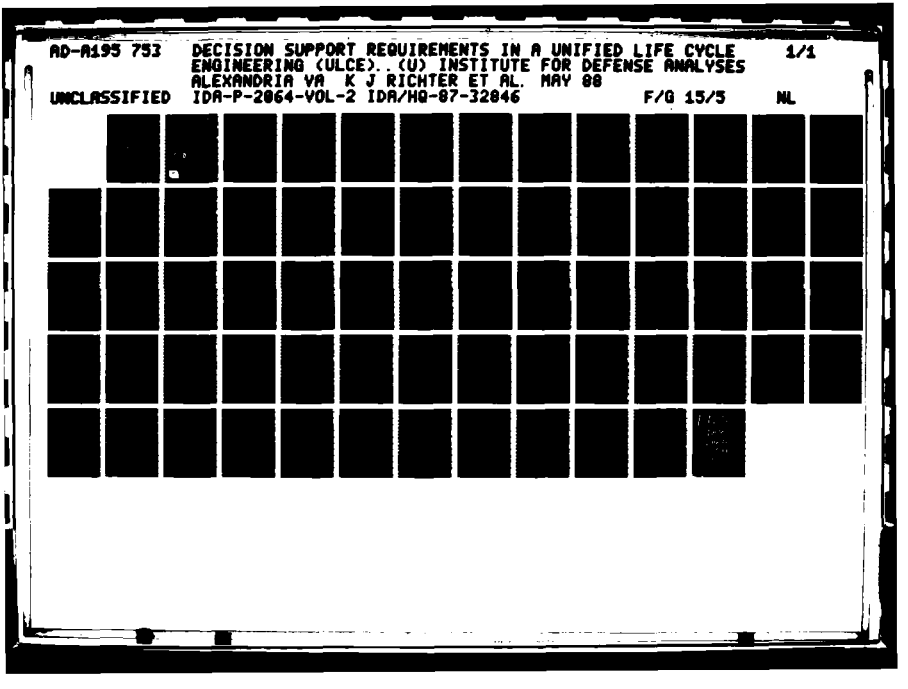
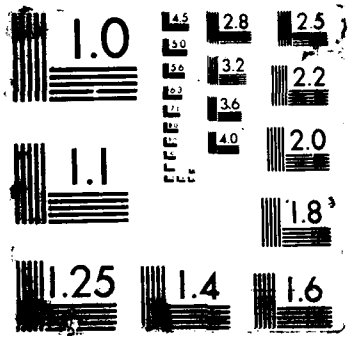
AD-A195 753 DECISION SUPPORT REQUIREMENTS IN A UNIFIED LIFE CYCLE ENGINEERING (ULCE).. (U) INSTITUTE FOR DEFENSE ANALYSES ALEXANDRIA VA K J RICHTER ET AL. MAY 88 1/1

UNCLASSIFIED IDA-P-2064-VOL-2 IDA/HQ-87-32046 F/G 15/5 NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3

(2)

IDA PAPER P-2064

DECISION SUPPORT REQUIREMENTS IN A UNIFIED LIFE CYCLE ENGINEERING (ULCE) ENVIRONMENT

Volume II: CONCEPTUAL APPROACHES TO OPTIMIZATION

Shapour Azam, University of Maryland
Joseph Naft, University of Maryland,
Michael Pecht, University of Maryland
Karen J. Richter, IDA

May 1988

Prepared for
Office of the Under Secretary of Defense of Acquisition
(Research and Advanced Technology)

Supported by
Air Force Human Resources Laboratory
Wright-Patterson AFB, Ohio

INSTITUTE FOR DEFENSE ANALYSES
1801 N. Beauregard Street, Alexandria, Virginia 22311

DEFINITIONS

IDA publishes the following documents to report the results of its work.

Reports

Reports are the most authoritative and most carefully considered products IDA publishes. They normally embody results of major projects which (a) have a direct bearing on decisions affecting major programs, or (b) address issues of significant concern to the Executive Branch, the Congress and/or the public, or (c) address issues that have significant economic implications. IDA Reports are reviewed by outside panels of experts to ensure their high quality and relevance to the problems studied, and they are released by the President of IDA.

Papers

Papers normally address relatively restricted technical or policy issues. They communicate the results of special analyses, interim reports or phases of a task, ad hoc or quick reaction work. Papers are reviewed to ensure that they meet standards similar to those expected of refereed papers in professional journals.

Memorandum Reports

IDA Memorandum Reports are used for the convenience of the sponsors or the analysts to record substantive work done in quick reaction studies and major interactive technical support activities; to make available preliminary and tentative results of analyses or of working group and panel activities; to forward information that is essentially unanalyzed and unevaluated; or to make a record of conferences, meetings, or briefings, or of data developed in the course of an investigation. Review of Memorandum Reports is suited to their content and intended use.

The results of IDA work are also conveyed by briefings and informal memoranda to sponsors and others designated by the sponsors, when appropriate.

The work reported in this document was conducted under contract MDA 903 84 C 0031 for the Department of Defense. The publication of this IDA document does not indicate endorsement by the Department of Defense, nor should the contents be construed as reflecting the official position of that agency.

This paper has been reviewed by IDA to assure that it meets high standards of thoroughness, objectivity, and sound analytical methodology and that the conclusions stem from the methodology.

ADA195753

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY DD Form 254 dated 1 October 1983	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) IDA Paper P-2064	5. MONITORING ORGANIZATION REPORT NUMBER (S)
6a. NAME OF PERFORMING ORGANIZATION Institute for Defense Analyses 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION OSD, OUSDRE, DoD-IDA Management Office
6c. ADDRESS (CITY, STATE, AND ZIP CODE) 1801 North Beauregard Street Alexandria, Virginia 22311	7b. ADDRESS (CITY, STATE, AND ZIP CODE) 1801 North Beauregard Street Alexandria, Virginia 22311
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Office of the Under Secretary of Defense (R&AT)/ET 8b. OFFICE SYMBOL	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER MDA 903 84 C 0031
8c. ADDRESS (City, State, and Zip Code) The Pentagon Washington, DC 20301-3080	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT / PROJECT NO. / TASK NO. T-D6-489 / ACCESSION NO. WORK UNIT

11. TITLE (Include Security Classification)
DECISION SUPPORT REQUIREMENTS IN A UNIFIED LIFE CYCLE ENGINEERING (ULCE) ENVIRONMENT
Volume II. Conceptual Approaches to Optimization

12. PERSONAL AUTHOR(S).
Karen Richter, Shapour Azarm, Joseph Naft, Michael Pecht

13. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final	FROM 3/87 TO 12/87	May 1988	63

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Decision Support System, DSS, Unified Life Cycle Engineering, ULCE, Optimization Software, Design Process, Decomposition, Measurement, Trade-offs, Electronics Design Optimization

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The goal of Unified Life Cycle Engineering is to develop an advanced design environment that allows considerations of producibility and supportability to be integrated into the design process in a timely fashion, i.e., early in the design process, along with the usual considerations of performance, cost, and schedule. A key factor in being able to develop an ULCE design environment is the management of the design decision-making process in such a way as to ensure that designs, optimized among all the competing factors, both up front and downstream, can be produced. This report is the result of a study addressing the design decision support problem under ULCE.

Volume I contains the results of the ULCE DSS Working Group efforts to outline a research and development plan for the design decision support for ULCE. Volume II is a review of the optimization techniques currently used or proposed to aid in the decision processes involved with competing requirements. The third volume of the report applies some of these optimization methods and other decision support techniques to actual design problems.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION	
☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS REPORT ☐ DTIC USERS	UNCLASSIFIED	
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code)	22c. OFFICE SYMBOL

DD FORM 1473, 84 MAR

C.S. 6-13-88

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

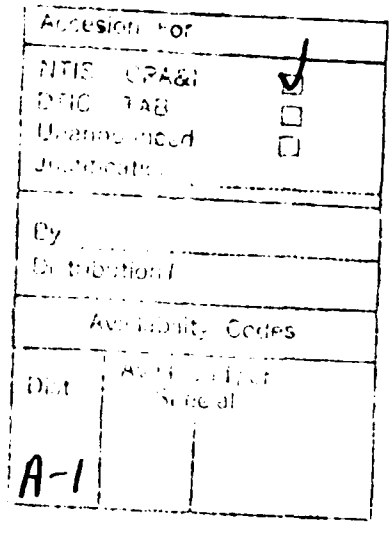
IDA PAPER P-2064

DECISION SUPPORT REQUIREMENTS IN A UNIFIED LIFE CYCLE ENGINEERING (ULCE) ENVIRONMENT

Volume II: CONCEPTUAL APPROACHES TO OPTIMIZATION

Shapour Azam, University of Maryland
Joseph Naft, University of Maryland,
Michael Pecht, University of Maryland
Karen J. Richter, IDA

May 1988



IDA
INSTITUTE FOR DEFENSE ANALYSES

Contract MDA 903 84 C 0031
Task T-D6-489

PREFACE

This report was prepared by the Institute for Defense Analyses (IDA) for the Office of the Under Secretary of Defense for Acquisition and the Air Force Human Resources Laboratory under contract number MDA 903 85 C 0031, Task Order T-D6-489, Decision Support Requirements - ULCE.

The issurance of this report meets the specific tasks of "[identifying] conceptual approaches to optimization among competing design requirements, [identifying] fertile areas of research which address the problems of design optimization and trade-offs in an ULCE environment, [and preparing] a plan for future research."

This report was reviewed by Drs. Jeffrey H. Grotte and Robert I. Winner of IDA and by Dr. Edison Tse of Stanford University.

CONTENTS

Appendices

ACRONYMS AND ABBREVIATIONS

AI	Artificial Intelligence
COAL	Committee on Algorithms
GAMS	General Algebraic Modeling System
GRG	Generalized Reduced Gradient
MOLP	Multiobjective Linear Programming
NLP	Nonlinear Programming
SLP	Successive Linear Programming
SQP	Successive Quadratic Programming
w.r.t.	With respect to

A. INTRODUCTION

Various tradeoffs among competing design requirements and goals will be encountered in Unified Life Cycle Engineering (ULCE). This second volume of the report, Decision Support Requirements in a Unified Life Cycle Engineering (ULCE) Environment, looks at the optimization techniques that will be needed by the design team in order to make decisions among these competing requirements and goals.

After the conceptual approaches to optimization are examined, the report will focus on the requirements needed to select a particular optimization method and to assess the results of that method. A list of currently available optimization software is included. In the concluding section, the research needs for design optimization as an integrated process for systematizing the tradeoffs in design will be identified.

B. PROBLEM DEFINITION

1. What is Design Optimization?

The design process generally consists of conception, invention, analysis, and refinement. Inherent in this process is the test and selection of a "best" design from among alternative designs. This is the main goal in design optimization.

In order to quantify this goal, the following questions are posed [Ref. 1]:

(1) How do we describe different designs?

(2) What is our criterion for "best" design?

(3) What are the "available means"?

The first question is realized in part by predictive design modeling. Design models differ from analysis models in that analysis models are generally developed based on the principles of engineering science, while design models are constructed for specific prediction tasks. For example, an analysis model may be developed for a circuit board in which we may compute its reliability based on the junction temperature of the components. One use of such a model may be to determine quantities such as the flow rate for convectively cooled boards as a function of the inlet temperature. Design models are quite different in nature. For example, we might design a circuit board to determine the routing and location of components while ensuring that it does not fail prior to a specified service

time. There is no single, unique design model, but rather different design models for different requirements.

The implication of the second question is that a design can be modified in order to generate different alternatives, and that there exists a goal that can be selected to give the most desirable alternative. Rational choice requires a criterion by which the different alternatives are evaluated and placed in some form of ranking. The choice of the criterion will be influenced by factors such as the design application, timing, the judgment of the designer, etc. Thus, the criterion quantity is a subjective and relative approximation of reality that is useful within the limitation of the model assumptions.

The third question is based on the fact that we are living, working, and designing in a world that is subject to the limitations imposed by the natural laws, availability of material, geometric compatibility, etc. Thus by "available" we signify a set of requirements that must be satisfied by an acceptable design. These requirements may not be uniquely defined, but are under the same limitations as the choice of our criterion. In summary, a design optimization can be defined to be the *best feasible rational* design. In order to be *best*, a desired objective should be minimized or maximized. In order to be *feasible*, a set of requirements should be satisfied. In order to be *rational*, all the required information and knowledge about what is to be optimized should be known, so an appropriate mathematical model can be presented.

2. The Design Optimization Model

The design optimization model involves a set of variables that describe the design alternatives, an objective (criterion) expressed in terms of design variables that we seek to minimize or maximize, and a set of constraints expressed in terms of the design variables that must be satisfied by any acceptable design. Formally, we assemble all the design variables in a vector $\mathbf{x} = (x_1,...,x_n)$. The sclar-valued objective function should be quantifiably expressed in terms of the design variables, i.e., $f(\mathbf{x})$, and the constraints expressed by vector-valued functional relations such as

$$\mathbf{h}(\mathbf{x}) = 0 \quad \text{and} \quad \mathbf{g}(\mathbf{x}) \leq 0. \qquad \text{(Eq. 1)}$$

Therefore, the formal design optimization model can be expressed by

$$\text{minimize } \{f(\mathbf{x}) : \mathbf{h}(\mathbf{x}) = 0, \mathbf{g}(\mathbf{x}) \leq 0 \} \qquad \text{(Eq. 2)}$$

where f is the objective function and **h** and **g** are the system of (functional) constraints.

From a modeling point of view, the functions f, **h**, and g may be given as explicit algebraic expressions of the design vector **x**. Often they are derived directly from basic equations and laws of engineering science. However, basic engineering principles are not always capable of describing the problem completely, so use of empirical or experimental data must be available to establish the relationships through curve-fitting of data or other means. In addition, there exists a possibility that the functions may involve complex procedures that may be realized by computer simulation models rather than equations.

In general, more than one objective function is possible. These objectives may sometimes be competing so that some compromise is required. The mathematical tools necessary for handling multi-objective models are rather specialized and not particularly useful for practical design problems. Multiobjective strategies will be presented in Section C.6.

"It is not always easy to decide whether a design characteristic should be associated with the objective or with constraint functions." [Ref. 2] For example, in the design of a circuit board a design engineer may be confronted with the problem of which design characteristics, such as cost, reliability, routing, etc., should be made the objective of the model. The reply might well be "none of them," as long as they all meet certain design specifications. However, if a description of how these characteristics affect the overall performance is considered, then one may begin to question the rigidity of the specifications, or may decide that they all should be considered in the objective function of the model.

3. Classification of Design Optimization Models

The vast majority of the design optimization models can be described in the standard form of Equation 2. The existence of such a standard form does not imply that all distinctions between models should be ignored. When faced with a design optimization model, it is usually advantageous to determine the special characteristics that allow the model to be solved more efficiently. The most extreme form of classification is to assign every model to a separate category. However, this approach would be based on the false premise that every difference is significant with respect to solving the model. A reasonable classification of design optimization models is as follows:

(1) Classification based on the mathematical characteristics. A classification scheme based on the dimensions of the design space and the nature of the functions has a significant algorithmic advantage. A typical classification is

Properties of **x**: one-deminsional, multi-dimensional

Properties of $f(\mathbf{x})$: continuously differentiable, twice-continuously differentiable, linear, nonlinear.

Properties of $\mathbf{h}(\mathbf{x})$ and $\mathbf{g}(\mathbf{x})$ (constraints): no constraints, linear, nonlinear, continuously differentiable, twice continuously differentiable.

(2) Classification based on the nature of design variables, which includes discrete, continuous, deterministic, and stochastic variables.

4. Feasibility and Critical Design Requirements

A problem that has been well-posed will have a solution and most often it will have more than one feasible solution, so that one may be selected as the optimum. A feasible solution is one that satisfies the constraints of the design optimization model. A problem is said to be well-posed in this context if it implies that the solution makes sense from an engineering point of view. Some possible difficulties in formulating a well-posed problem are the emptiness of the feasible domain, which implies that no acceptable design exists, or unboundedness of variables, which implies that no finite values of the design variables that make sense from an engineering design point of view yield an optimal solution to the model.

Once the feasibility of the model is addressed, the next task is to identify the critical design constraints. Loosely speaking, an active or critical constraint is one that would change the location of the optimum if removed from the model. For inequality constraints this means that they must be satisfied as a strict equality at the optimum. A critical constraint (or design requirement) is also referred to as an active constraint. Identification of active constraints is important in design optimization because they often are associated with the failure modes of the design. In fact, some traditional design methods were really primitive design optimization in that failure modes were considered critical a priori. This was usually done relative to some often hidden criterion. Essentially, a design problem was solved by assembling enough active constraints to make a system of n equations of the n unknowns of the design variables [Ref. 1].

A non-critical or inactive constraint is defined as one whose presence defines the feasible domain but that plays no role in the location of the optimum.

5. Multistage vs. Multilevel Decomposition

The successful design optimization of complex systems invariably involves decomposition of the system into a number of smaller subsystems, each with its own goals and constraints [Ref. 3]. The resulting interconnections of the subsystems generally take the form of a multistage or multilevel decomposition. In the multistage decomposition, design decisions are made sequentially at different stages so that the output of one stage is the input of the succeeding stage. If the number of stages tends to infinity, the problem becomes an infinite stage or continuous problem. In multilevel decomposition, the interconnection of subsystems is of hierarchical form. In this approach, a given-level unit controls or coordinates the units on the level below it and, in turn, is controlled by the units above it.

Dynamic programming is a mathematical technique well suited for optimization of multistage models. This technique was developed by Richard Bellman in the early 1950's [Ref. 4]. The dynamic programming technique decomposes a multistage design problem into a sequence of single-stage design problems. Thus, an n-variable problem is transformed into a sequence of n, single-variable problems that are solved successively. The decomposition to n-subproblems is done in such a manner that the optimal solution of the original n-variable problem can be obtained from the optimal solutions of the n one-dimensional problems.

A different approach has recently been proposed by Johnson and Benson [Refs. 5, 6], where a design optimization model is decomposed into several modular components and an integrating component. Modular components are solved sequentially and exactly. The solutions of the modular components are then inserted into the integrating component to obtain a solution that will be fed again into the modular components. This process is repeated until it converges to a solution.

The basic steps of multilevel decomposition in design optimization are very similar to the Dantzig-Wolfe approach [Ref. 1]. The method has a multilevel structure that breaks down a problem into several smaller subproblems which can be solved independently and then coordinates each solution to obtain an optimal solution. A number of papers and several books have been published on the subject of multilevel decomposition methods (see for example, Lasdon [Ref. 8]; Wismer [Ref. 3]). Essentially, these methods are combinations of the model coordination method and the goal coordination method.

In the model coordination method, the decomposition is made possible by adding constraints to the mathematical model of the problem in the form of fixing variables in order to coordinate the activities of the subproblems. The model coordination method is also known as the feasible decomposition method due to feasibility of intermediate values of design variables. The method is particularly attractive from an engineering design point of view since the iteration process may be terminated whenever it is desired with a feasible, though nonoptimal, design [Ref. 9].

In the goal coordination method, decomposition is made possible by modification of the objective (goal) of the subproblems while cutting design variable links between subproblems. This method is also known as the dual method since the upper-level problem is the dual of the lower-level subproblems [Ref. 8]. A shortcoming of this method from an engineering design point of view is the infeasibility of intermediate values of the design variables. The iteration process must proceed until the optimum is reached.

6. Trade-off Curves and Sensitivity Considerations

Every design has a set of design characteristics that can be thought of as generating a corresponding set of values. Thus, once a specific configuration is available, the designer can set down a list of significant design characteristics and a list of significant values. It frequently occurs in practice that there is more than one design characteristic that generates values for the designer. They commonly oppose each other in the sense that if one is changed to increase the design value, the other must be changed to reduce the value. The choice of the best design is a trade-off between these design characteristics. For a given general configuration, a trade-off curve can be developed that is a locus of all optimum designs such that each point on the curve is a trade-off point [Ref. 2]. Trade-off curves also can be thought of as a plot of optimum designs corresponding to variations in design parameters or assumptions. An analysis using these curves is called a sensitivity analysis. Since this type of information is so important in implementing a solution on a real system, in many cases, a detailed sensitivity analysis is more valuable than the actual optimal solution itself.

The reasons for performing a detailed sensitivity analysis are given by Reklaitis, Ravindray and Ragsdell [Ref. 10]:

1. To determine the parameters to which the optimal solution is sensitive. If such parameters exist, then it may be advantageous to modify the associated system features. For example, if it is discovered that the solution is sensitive to the

availability of heat sinks (a constraint placed on the system in circuit board design), then it may be necessary to increase the number of available heat sinks.

2. To extract information about additions or modifications to the system for improving the overall operation. Thus, one can obtain information on the advisability of adding new production capacity or increasing intermediate storage.

3. To clarify the effect on the system of variations in imprecisely known parameters. Some model parameters may be subject to uncertainty. Sensitivity analysis can indicate whether it is advantageous to allocate resources to obtain better estimates of these parameter values. Alternatively, parameters that initially appear to be critical may turn out to be unimportant and may not need further refinement.

4. To suggest the effects of variations in uncontrollable external parameters. For example, system inputs such as product demands may be outside the control of the designer. Parameter sensitivity analysis can estimate the effects of product demands on profits and hence allow the user to plan for a range of economic returns.

C. CONCEPTUAL METHODS

This section begins with a discussion of the classical methods that are useful in finding the optimum of continuous or differentiable functions of real variables. Under this heading, optimality conditions for single-variable functions, and the extension of the conditions to the multivariate case, are discussed. The next topic considered is unconstrained methods, which are important because some methods of solving constrained optimization problems are based on transforming these problems to unconstrained problems (indirect methods). This is followed by a discussion of constrained optimization problems, beginning with the linear case (linear programming methods and integer programming) and continuing to discuss nonlinearly constrained problems, including geometric programming Monotonicity analysis, multi-objective methods and multilevel decomposition methods are then discussed, concluding with a section on artificial intelligence (AI) related techniques.

The numerical methods presented here have gone through extensive analysis and/or testing in the literature. While much research in design optimization remains to be done, the field has matured to the point where the techniques can be applied to a large percentage of design tasks. Numerical optimization provides us with a new design philosophy in that

a systematic approach to design decisions is provided; however, this should not suggest that designer's intuition and experience are unimportant. In fact, in order to establish a true design environment, the designer should be kept in the design loop (computer-aided) using interaction and other means.

In presenting the materials of this section, extensive citations to various articles and/or books is given to avoid (as much as possible) mathematical and/or theoretical discussions regarding various methods. Furthermore, attempts have been made to present the subject from the pragmatic viewpoint of a design engineer.

1. Classical Methods (Optimality Conditions)

Here, "classical methods of optimization" means those which are analytical in nature and use differential calculus techniques in locating the optimum points. In order to apply these methods to any problem, the corresponding functions should have certain properties such as continuity and differentiability (see Apostol [Ref. 11]). Although the classical methods have limited applications, the study of these methods forms the basis for developing most of the numerical methods presented in the subsequent sections.

The classical methods are basically a set of necessary and sufficient conditions (optimality conditions) that should be satisfied by an optimum in order to be accepted as a solution. The treatment of some of these methods goes back several centuries, hence the name classical.

Fermat was the first person to propose the optimality conditions for single-variable functions during the seventeenth century. Although Fermat's method seemed to be logically inconsistent, it was extended by Euler to multivariable functions and later by Lagrange, who contributed greatly to optimization theory. Fermat's contradiction was finally removed when Cauchy defined the concept of "limit" [Ref. 12].

In the last 50 years, substantial progress in the theory of optimization has been made. The most important advance gives the optimality conditions for constrained problems, the so-called Karush-Kuhn-Tucker [Refs. 13, 14] conditions.

The necessary conditions of optimality refer to those conditions that an optimum point must satisfy. However, it is possible that a nonoptimal point also satisfies the necessary conditions. For this reason, a point that satisfies the necessary conditions is often called a stationary point. The nature (maximum, minimum, etc.) of a stationary point is determined then using the sufficient optimality conditions.

Several topics on the question of optimality conditions can be found in the literature [Refs. 15, 16], each topic being characterized by the assumptions made on the functions involved. The optimality conditions stated by most of these topics relate in one way or another to the concept of *Lagrangian*, which can most conveniently be treated, without the loss of much generality, if the objective and constraint functions are differentiable. Relaxing the differentiability assumptions usually leads to optimality conditions that can best be expressed as optimality conditions on some other problem, such as finding a saddlepoint of a Lagragian or solving a so-called dual problem.

It should be noted that in the design optimization problem, the most often sought solution is the global solution. If the problem has more than one local optimum (local solution) the global solution often requires determining all the local optima, evaluating the objective function at each optimum, and choosing the global optimum.[1] (Most of the theory and computational algoithms available are for determining local optima, unless the problem has only one local optimum in which case the local and the global opitmum are the same.) However, if all of the local optima cannot be determined, despite all of the efforts, then the global optimum could be missed.

In the following sections, a selection of many methods available for various problem categories is described. The description is intended to present an overview of those methods that are popular. The cited references may be consulted for further information about those, and other, methods.

2. Unconstrained Methods

a. One-Dimensional Methods

An unconstrained design optimization problem in which the objective function is a function of only one variable is the most elementary type of problem. Yet, it is of central importance, not only because it is encountered very often in engineering design problems, but also because it arises commonly as a subproblem within the iterative procedure of multivariable optimization problems [Ref. 10].

The one-dimensional numerical methods described in the literature may be classified into three categories: (1) region elimination methods, (2) polynomial approximation

---

[1] For certain functions, such as convex functions, all local optima are also global optima.

methods, and (3) derivative-based methods. All of these methods require that within the domain of interest the objective function be unimodal. When this happens, the objective function has a unique minimum within the domain of interest. Of course, in many engineering design problems the unimodality assumption does not hold true, and, in any case, it cannot be easily verified. One way to handle this difficulty, especially if the initial domain of interest is large, is to divide it into smaller intervals, find the minimum over each subinterval, and then select the smallest of the minima over the subintervals.

The region elimination methods are based on successive elimination of regions where the optimum is not located. In general, these methods have two phases [Ref. 10].

1. The Bounding Phase is an initial coarse search that will bound the optimum.
2. The Interval Refinement Phase is a finite sequence of interval reductions to reduce the initial search interval to a desired accuracy.

Several region elimination methods have been discussed in the literature, including the Golden Section, Fibonacci, Dichotomos, and Uniform methods [Ref. 16]. These methods do not use the information about the derivative [Ref. 11] of the function. Hence, they are applicable to both continuous and discontinuous functions and to discrete variable problems.

Polynomial approximation methods require that the functions be sufficiently "smooth" (see, for example, Apostol [Ref. 11]). The basic motivation is that if the function is smooth, it then can be approximated by a polynomial, and the approximating polynomial can be used to predict the location of the optimum. The only requirement for these methods to be effective is that the function in question be continuous and unimodal. The polynomial approximation usually is in the form of a quadratic, such as in Powell's method [Ref. 17] which is a successive quadratic approximation method and Davidon's method [Ref. 18] which is a cubic interpolation method.

If the objective function in question is differentiable, then further efficiencies in the search method for the optimum could be achieved using the derivative-based methods such as Newton-Raphson, Bisection, and Secant, and the quadratic and cubic approximation methods [Ref. 10].

It should be noted that from a theoretical point of view, the polynomial approximation methods (derivative and nonderivative-based), such as quadratic and cubic search methods, are superior to region elimination methods. This claim seems to be supported by the limited computational experiments presented by Himmelblau [Ref. 19].

b. Multidimensional Methods

Many engineering design problems require that the minimum of some function of several variables is found where no restrictions are imposed on the variables [Ref. 20]. Methods for solving this type of problem are the subject of this section. These methods in general have three components [Ref. 16]:

(1) Find the direction of search for the optimum.

(2) Move in this direction as much as improvement in the objective function is possible (one-dimensional search).

(3) Determine when the process has converged to an acceptable solution.

Methods for multidimensional, unconstrained design-optimization problems are classified according to their need for derivative information about the function to be optimized. They are classified into: (1) zero-order methods, (2) first-order methods, and (3) second-order methods [Ref. 20].

The zero-order methods are those that require function values only to obtain the optimum. They are generally easy to program and reliable in practice for the general types of functions encountered in design problems. However, the price paid for this generality is that these methods often require numerous function evaluations to reach the optimum. Hence, they are not useful for problems when the function evaluations are computationally expensive. Among the methods in this category are [Refs. 20, 16]: the Complex method and the methods of Rosenbrock, Powell, Box, and Hooke and Jeeves. This short list is by no means exhaustive, but serves to indicate the existence of a variety of algorithms for the zero-order methods.

First-order methods use derivative information to find a direction toward the optimum. These methods should perform better than the zero-order methods simply because the designer is provided with more information on which to base the optimization decisions. One of the oldest of these methods is the one credited to Cauchy [Ref. 21]. This method, which is also called the steepest descent method, usually does not perform well for some design problems as it approaches the optimum [Ref. 16]. The conjugate direction of Fletcher and Reeves [Ref. 22] requires a simple modification to the steepest descent method and yet dramatically improves the efficiency of the optimization process. Perhaps the most efficient methods among the first-order methods are the so called variable metric methods, such as Davidon-Fletcher-Powell or DFP [Refs. 23, 24] and Broyden-Fletcher-Goldfarb-Shanno or BFGS [Refs. 25 through 28]. Currently, there is a

significant amount of work underway to use the variable metric methods in constrained and even large-scale problems. This research will be further discussed in Section F.1 of this report.

In the second-order methods, the matrix of second derivatives of the function, which is called the Hessian matrix, is used for the search toward the optimum. Newton's method together with its various modifications are among the most effective methods in this category. The principal difficulty with some of these methods, despite their high performance when applicable, is that they may have a singular and indefinite Hessian matrix. Further, they may not have the general descent property, which is desirable for these types of methods [Ref. 10].

Numerical experiments published in the literature suggest that BFGS, DFP, Powell's method, and various modifications of Newton's method are among the most successful multidimensional methods available in the literature [Refs. 18, 19, 29].

3. Linearly Constrained Methods

Linearly constrained problems are those in which the functions **h** and **g** of Eq. 1 are linear functions of the design variables. Some basic methods of solution for these types of problems are given below.

a. Linear Programming Methods

The most thoroughly developed and understood optimization problem is the linear programming problem. However, most engineering problems of practical interest are not of this form. Therefore, in numerical optimization the study of linear programming is usually overlooked in favor of the methods that are applicable directly to nonlinear problems. However, an understanding of linear methods is important for two reasons. First, it may be possible to simplify a nonlinear problem into a linear one and then use the linear programming methods to solve it. Second, linear programming methods are often used as the basis of the more complex nonlinear programming methods.

"The most common method for the solution of the linear programming problems is referred to as the Simplex method. This method was developed by Dantzig in the late 1940s [Ref. 30]. Computer codes based on this method are available on most computer systems. These have usually been tested extensively and are highly reliable. The Simplex method has been used to solve a number of military, economic, industrial, and societal

problems" [Ref. 20]. In fact, some of the programs written now can handle problems consisting of several thousands of variables and constraints quite efficiently. In a 1976 survey of American companies, the Simplex method came out as the most often used technique (74 percent) among all optimization methods [Ref. 31]. About one-fourth of the computer time spent in 1979 on scientific computing was devoted to solving linear programming problems [Ref. 32].

The Simplex method operates on the boundary of the feasible domain by moving from one vertex to the other while at the same time improving the objective function. Roughly speaking, the complexity of the edges and vertices that define the feasible domain grows exponentially with the size of the problem. For this reason, the Simplex method is an exponential method in that the number of iterations required to reach an optimal solution is an exponential function of the number of variables and constraints. Hence, the insight that an algorithm operating in the interior of the feasible domain should avoid such complexity was used by Karmarkar to develop a new and in some cases much more efficient method of linear programming [Ref. 33].

b. Integer Programming

In all the methods considered so far, each of the design variables is permitted to take any real (or fractional) value; however, there are certain practical problems in which the fractional values of the design variables are neither practical nor physically meaningful. If an integer solution is desired, it is possible to use any of the techniques described previously and to round-off the optimum values of the design variables to the nearest integer values. However, in many situations it is very difficult to round-off the solution without violating any of the constraints. Frequently, the rounding of certain variables requires substantial changes in the values of some other variables in order to satisfy all of the constraints. Furthermore, the round-off solution may give a value of the objective function that is very far from the original optimum value. All these difficulties can be avoided if the optimization problem is posed and solved as an integer programming problem.

When all of the variables are constrained to take only integer values in an optimization problem, it is called an (all) integer programming problem. When only some variables are allowed to take integer values, the optimization problem is called a mixed integer programming problem. When all the design variables of the optimization problem

are allowed to take on values of either zero or one, the problem is called a zero-one programming problem.

Among various methods available for solving the all-integer and mixed-integer linear programming problem, the cutting plane method of Gomory [Ref. 45] and the branch and bound method of Land and Doig [Ref. 46] have been very popular. An efficient and interesting enumerative method was also developed for zero-one linear programming problems [Ref. 47]. Very little work has been done in the field of nonlinear integer programming. One approach suggested in the literature is the so-called generalized penalty function method [Refs. 48, 49].

4. Nonlinearly Constrained Methods

This section deals with techniques that are applicable to the solution of general nonlinearly constrained design optimization problems. The methods can be divided into two broad categories--the indirect and the direct methods. In the indirect methods, the original constrained problem is transformed into a sequence of unconstrained optimization problems. In the direct methods, on the other hand, the constraints are handled in an explicit manner. Geometric programming for posynominals is also covered.

a. Indirect Methods

The idea of converting a constrained optimization problem into a sequence of unconstrained problems is very appealing, since unconstrained problems can be solved both efficiently and reliably. Of course, it is hoped that only a few unconstrained subproblems of moderate difficulty will be required to approximate the constrained solution with acceptable accuracy. Three traditional approaches in this category are: (1) the exterior penalty function method, which penalizes the objective function only when the constraints are violated; (2) the interior penalty function method, which penalizes the objective function as the design approaches a constraint from inside the feasible domain; and (3) a combination of the interior and exterior penalty methods.

Barrier function methods are also used to transform a constrained problem into a sequence of unconstrained problems by setting a barrier on the design point so that it cannot leave the feasible domain [Ref. 16].

While the aforementioned methods are easy to implement, the ill-conditioning involved in the penalty or barrier type of methods limits their utility for practical design problems. However, by using information about Lagrange multipliers within these

methods, their performance can be substantially improved. In fact, Powell [Ref. 34] notes that the use of penalty-type methods, which do not use the Lagrange multipliers, is obsolete in practical optimization. Therefore, the augmented Lagrangian methods (or methods of multipliers) that use the information about the Lagrange multipliers are considered to be substantially superior to the penalty and/or barrier methods [Refs. 20, 16].

b. Direct Methods

In this section, those methods that deal directly with the constraints are described. Methods in this category are: (1) heuristic search methods, which are intuitive and do not have much theoretical support, such as the random search and complex methods; (2) constraint approximation methods, which approximate the objective function by quadratic or linear functions and the constraints by linear functions; and (3) the feasible direction methods that produce an improving succession of feasible designs. Attention here will be focused on those methods that fall in classes (2) and (3).

Among the most efficient methods in class (2) are the sequential linear programming (SLP) and sequential quadratic programming (SQP) methods. In the sequential linear programming, linear approximations to the objective and constraints are made about the current point. Then, the resulting linear programming problem is solved. The process is repeated successively, until the solution to the original problem is obtained. The method also is referred to as Kelley's cutting method [Ref. 35]. In the sequential quadratic programming method, the objective function is approximated by a quadratic function and the constraints by linear functions at the current point. The resulting problem (a quadratic programming problem [Ref. 36]) is then solved by finding a search direction. Having determined the search direction, the current design is updated by minimizing a combination of objective function and current constraints' violation [Ref. 37]. Currently, SLP and SQP are considered among the most successful algorithms in nonlinearly constrained design optimization methods [Refs. 38, 39].

In class (3), the feasible direction methods, the nonlinearity of the design problem is dealt with directly. A subclass of these methods is the gradient projection methods [Ref. 40]. One of the main advantages of these methods from an engineering design point of view is that the generated intermediate design point is feasible. Hence, the iteration process may be terminated whenever desired with a nonoptimal, but acceptable, design.

The method of feasible direction was first presented by Zoutendijk [Ref. 41] and modified by various people, including Topkis and Veinott [Ref. 42], to improve its

convergence properties. Another method in this class depends upon reducing the dimension of the design problem by representing some design variables in terms of an independent subset of the rest of the variables. This is the reduced gradient method, which was first developed by Wolfe [Ref. 43]. The method was later generalized (generalized reduced gradient, GRG) by Abadie and Carpentier [Ref. 44]. In GRG the inequality constraints are handled either by explicitly writing these constraints as equalities or by the concept of active set strategy (see, for example, Gill et al. [Ref. 18]). If an active set strategy is employed, then at each iteration the active constraints must be estimated and their linearization added to those of the equalities. Proper steps in case of infeasibilities should be included to locate a point satisfying both active and inactive constraints. Wilde and Beightler also developed their differential algorithm based on the constrained derivatives. [Ref. 12] This method employs much the same theoretical basis as the GRG method.

c. Geometric Programming

Geometric programming is a relatively new method for solving a particular type of nonlinear programming problem. The method optimizes functions that can be expressed as posynomials subject to posynomial constraints. A posynomial is a polynomial with positive coefficients and variables and real exponents. Geometric programming is distinct from other optimization techniques in its emphasis on the relative magnitudes of the various terms that make up the objective function rather than the variables. Other methods find optimal values of the design variables first, whereas geometric programming first seeks the optimal value of the objective function. This has a particular advantage in cases where the optimal value of the objective function is all that is of interest, i.e., where the explicit calculation of the optimal design vector may be omitted. Furthermore, geometric programming can often reduce a complex optimization problem to a set of simultaneous linear equations. On the downside, the primary drawback of geometric programming is that it does require that the problem can be formulated in terms of posynomials.

The basis of geometric programming is the general arithmetic mean--geometric mean inequality (sometimes called Cauchy's inequality), which states that for any n nonnegative numbers $x_1, x_2,...,x_n$

$$\frac{x_1 + x_2 + ... + x_n}{n} \geq (x_{1*}x_{2*}...x_n)^{1/n}$$

with the equality holding only if $x_1=x_2=...=x_n$. Using the inequality, the objective function is rewritten and becomes the subject of the inequality. The original objective function, having a form similar to the left-hand side of the inequality, is then called the

primal function. Its transformed version on the right-hand side is called the dual function. The result is that the maximum of the dual function equals the minimum of the primal function. The optimization proceeds by minimizing the primal or maximizing the dual, whichever is easier. Furthermore, the maximization of the dual, subject to orthogonality and normality conditions, is a sufficient condition for the primal function to be a global minimum.

5. Monotonicity Analysis

For a design engineer who is examining a problem, the location of the optimum is only one of the goals to be attained. Specific insight, possibly provided to the designer, is another goal. Therefore, it is important to develop a clear understanding of the critical design requirements (active constraints). Monotonicity analysis is an optimization technique that provides such information. It may be applied to design problems with monotonic properties--a situation very common in design problems [Ref. 50]. In such problems, the objective and constraint functions are monotonic with respect to a variable in a certain range of that variable. Based on certain necessary rules derived at the optimum, the designer may be able to detect which constraints are critical (active). For inequality constraints, this means that they should be satisfied as strict equalities at the optimum. Information about activity and/or inactivity of the constraints is important for a design engineer because it can be used to identify what gains could be achieved if the boundary of the feasible domain were modified, which in turn would point out directions for desirable technological improvement.

Monotonicity analysis was first presented by Wilde [Ref. 51]. The motivation for the method came from the fact that numerical optimization techniques may give only a limited insight into underlying design principles for a given problem and that there may be certain mathematical properties in design problems which otherwise may be exploited. Wilde later applied monotonicity analysis to a number of design problems to identify their global solution [Ref. 49]. Papalambros and Wilde [Refs. 1, 52, 53] have applied the monotonicity analysis to many engineering design problems. Implementation of the monotonicity analysis in a computer algorithm was first demonstrated by Zhou and Mayne [Ref. 54] in an interactive manner. A fully automated version of the method was developed by Azarm and Papalambros [Ref. 55] and Zhou and Mayne [Ref. 56].

6. Multiobjective Methods

a. Multiple Objective Problems

The complexity of engineering design situations dictates that there is often more than one objective. Alternative designs in these cases must be evaluated according to multiple criteria. For these reasons, extensions to single objective methods have been created to deal with the multiobjective problem. For Unified Life Cycle Engineering, the multiple objectives include cost, schedule, performance, producibility, supportability, and others.

The ideal way to solve a multiple objective problem would be to determine the decision maker's utility function U as a function of the various objective functions and then to solve the single objective optimization problem with U as the single objective. The difficulty with this approach is that of determining the utility function and its specific dependence on the objectives. The alternative is to search the space of tradeoffs among the individual objectives for an optimal solution. Since the trade-off space is generally very large, the practice has been to use interactive man-machine procedures to solve this type of problem.

A solution is optimal if it maximizes the utility function. In order to be optimal, its criterion vector (made up of the objective functions) must be non-dominated. That is, it must not be possible to find a feasible point that increases an objective without decreasing at least one other objective. A point in the set of feasible solutions is called efficient or Pareto optimal if it corresponds to a nondominated criterion vector. Near-optimal solutions may satisfy the decision maker. Any solution, whether optimal or near-optimal, that successfully terminates the decision process is called a final solution.

The following sections give brief overviews of several of the more important classes of methods for multiobjective optimization.

b. Multiobjective Linear Programming

One method in Multiobjective Linear Programming (MOLP) is the point estimate weighted sums approach. In this method, each objective is multiplied by a strictly positive scalar weight. Then the weighted objectives are summed to form a composite or weighted-sums objective function. The weighting vector is normalized so that its elements sum to one. Then the linear programming approach is used to maximize the composite objective function. From this, it is hoped that an optimal solution will be found, or one that is near

enough to the optimal to be satisfactory. The point estimate weighted-sums technique can be viewed as a method that varies convex combinations of the objectives.

Optimal weighting vectors are difficult to estimate because the set of optimal weighting vectors is a function of the decision maker's preferences, the relative lengths of the objective function gradients, and the geometry of the feasible region. Another problem with the use of weighting vectors is the correlation between objectives, which can cause suboptimal results. A further difficulty in MOLP is the need for consistent scaling of the objective functions. Three approaches to rescaling are (1) normalization, (2) using 10 raised to an appropriate power, and (3) the use of range equalization factors.

In MOLP, if one or more of the objectives are linear fractional,[2] the technique of Multiple Objective Linear Fractional Programming (MOLFP) is used.

c. Goal Programming

Goal programming is an important area in multiobjective optimization. The basic idea is to establish a goal level of achievement for each criterion. This method is particularly well suited to criteria for which threshold values of achievement are significant. Goal programming is different from linear programming in the following ways.

- Objectives are thought of as goals.
- Priorities and/or weights are assigned to the achievement of goals.
- Deviation variables are used to measure over- or under-achievement of the target values
- The weighted sum of the deviation variables is minimized to find the best solutions.

Rarely will one feasible point satisfy all the goals, so points are sought which satisfy all goals as closely as possible.

d. Interactive Procedures

One of the strongest areas of current development in multiobjective optimization is in interactive procedures in which the feasible region is explored for an optimal or near optimal solution. Interactive procedures alternate between phases of decision making and phases of computation. There is a feedback between the user and the model that enables

---

[2] An example of a linear fractional function is $g = \frac{ax+b}{cx+d}$.

the user to gain insight into the problem. These methods enable the user to make midcourse corrections to the solution search process.

There are a number of interactive procedures available. They can be generally classified as feasible region reduction, weighting vector space reduction, criterion cone contraction, and line search. Although very different in approach, these algorithms all show quite rapid convergence--in approximately n iterations, where n is the number of objectives.

7. Multilevel Decomposition Methods

Since the publication of Dantzig and Wolfe [Ref. 7], there have been numerous pieces of literature describing algorithmic development and/or application of decomposition in design optimization. More recently, many engineering problems have been solved using decomposition methods including those in mechanical [Refs. 5, 6, 57, 58]; structural [Refs. 9, 59]; and aerospace design [Refs. 60, 61].

There are several reasons why a multilevel decomposition method should be used to obtain the optimal solution to an engineering problem. First, many engineering problems are, by their nature, decomposable to several subproblems. For example, an aircraft is a complex system composed of several subsystems including the structure, engine, landing gear, etc. Using a two-level decomposition, each subsystem may be optimized at the first level, and then the subsystems' solutions are coordinated at the second level to obtain the optimal solution of the original problem which is the aircraft. Second, the interdisciplinary nature of decomposition methods forces even more interaction across disciplines. [Ref. 62]. For example, optimal design of an aircraft may be obtained by analysis of its engine, aerodynamics, structure, material, etc., independently and then these analyses may be integrated using a decomposition-based optimization. Third, it may be more effective to use different specialized optimization techniques for various subproblems. Finally, multilevel decomposition methods fit well into distributed or even parallel processing capabilities, which are typical of a modern computing environment.

The idea of decomposition for solving nonlinear systems (in particular, large-scale systems) was first proposed by Kron [Ref. 63]. He indicated that "physical systems with a very large number of variables (say, with tens of thousands) may be solved with available digital computers by tearing the system apart into a large number of small subsystems." However, as was mentioned before, it was the publication of Dantzig and Wolfe [Ref. 7] that initiated the extensive work on this method in design optimization.

Azarm and Li [Refs. 64, 65] have recently extended the model coordination method by coupling it with the global monotonicity analysis. In their method, the monotonicity analysis is used in the first level of a two-level decomposition method to identify the active constraints. This information is then sent to the second-level problem, which in turn finds a new point for an improved objective function. The new point is then sent back to the first-level subproblems, and the iteration process continues until an optimal design is obtained.

More on system-level optimization can be found in Section C of Volume II of this report.

8. Artificial Intelligence (AI) Methods in Design Optimization

a. Issues in Design Optimization

The use of AI optimization techniques in the design process is based on the premises that successful implementation of numerical algorithms requires the application of heuristics and that the design process can be improved by the use of heuristics based on knowledge captured during the iterative process [Refs. 66-70]. Associated with the first premise is a knowledge structure that represents attributes about classes of design problems and about optimization results accumulated over a period of design times. Associated with the second premise is a knowledge structure that generates decision support data during the iterative process of a particular design optimization problem.

No single optimization procedure can efficiently solve all classes of problems. However, by monitoring characteristic responses from an optimization procedure or from different classes of problems, rules can be developed to select the appropriate algorithms to solve a particular problem or part of a problem. For example, the degree of linearity (nonlinearity) and constraint information can be used to select the type of optimization approach to be applied. The selection of a starting point is another parameter which can be intelligently selected based on previous experience. Both the feasibility of the starting point and the closeness of a current design point to violating a constraint can be useful information. Details of other characteristics will be discussed later.

AI capabilities can also be included to examine the efficiency of an algorithm within the iterative design cycle. This examination may result in parameter adjustment or even in the selection of a new starting point, depending on the pattern (trend information) of the data during iteration. Often, an improper starting point converges very slowly (as noted by a fixed design status) to the goal region in the design space. A change in the convergence

criteria may also be necessary. This is an important consequence of the optimization technique employed, but it also can occur due to round-off, truncation, etc.

In general, it is possible to calculate data in a bounded region using global information. Data from each iteration is saved to provide knowledge about the problem, and it can aid in parameter selection. Since functions of the problem depend implicitly on design variables, the function evaluations are tedious and time consuming. Similarly, gradient evaluations are complex and generally require special purpose algorithms. In fact, during the design optimization cycle, most of the computational effort goes to function and gradient evaluation. Thus, checks to identify anomalies in the gradient calculation of cost and constraint add to the optimization efficiency.

Constraints also can be examined to determine whether they correspond to a feasible design solution. It is necessary that constraints that cannot be satisfied be identified early in the optimization planning stage.

b. Utilization of Knowledge in Optimization

Design optimization requires knowledge types that correspond to knowledge domains in the AI sense. An appreciation for this fact can be gained by recalling the decisions and tasks necessary during an optimization study.

Development of a good model requires engineering knowledge about the specific problem and mathematical knowledge about its properties and possible methods of solution. The following modeling tasks have been identified

- Examine the consistency, boundedness and feasibility of the model
- Define all useful transformations
- Develop procedures to handle discrete variables
- Develop procedures to handle scaled variables and constraints
- Check for redundant constraints
- Determine the mathematical form of the model
- Determine the need for parametric or sensitivity analysis.

Once the model has been selected, the development of a solution strategy must be considered. The following tasks have been identified:

- Determine the numerical methods
- Define program parameters (e.g., step size, accuracy, penalties, termination)

- Determine the global or local solution and the criteria for accepting a solution
- Determine the active constraints
- Define an error trapping, identification, and alternative methodology scheme.

All of these tasks can be addressed automatically (or at least interactively) with a computer. The answers may require a high degree of programmed expertise and computational analysis, but the knowledge to handle these questions generally exists.

It is apparent in engineering optimization that the analyst must be an expert both in the technology of the model and in the mathematics of the optimization techniques. AI offers a means to simplify the task, thereby encouraging wider and more correct use of optimization. From the AI viewpoint, two problem categories can then be identified: problem-solving techniques and knowledge engineering.

Problem-solving typically represents graph-type searches similar to those found in operations research. One approach is the production system that utilizes condition-action pairs, usually of the form: IF (precondition), THEN (action), commonly called production rules. A production system is characterized by a knowledge map, production rules, and a control strategy.

The problem-solving process is described by movements in a state space. The knowledge map contains information representing the possible states of a problem. A goal state is set to represent the desired result of the problem-solving process.

Production rules are the means by which a given state is changed to another one, closer to the solution. The rules are independent, but they all communicate to affect the knowledge map. As data is changed, the map reflects the change, and the system may be updated in the form of new rules.

The control strategy provides the order in which the rules are applied to the knowledge map. This typically leads to a heuristic graph search, the nodes being the various states.

The task of formulating and incorporating design rules as knowledge is complex. However, it is necessary to establish models for design rule development that can serve as a basis for the establishment of further rule development. Some simple rules have been listed in the research of Arora and Baenziger [Ref. 67]. The rules represented there pertain to active constraint instability, where there is conflict, in an attempt to move the search direction normal to the constraint in question while improving the objective function value.

Other representative production rules also have been identified by Li and Papalambros [Ref. 71].

c. A Methodology For Knowledge Processing

Optimization methods have been applied to many engineering design as well as operation research problems. When one or more of the objective or constraint functions is nonlinear, then, we have encountered a nonlinear programming (NLP) problem. The solution strategies for NLP problems generally utilize techniques that are based on local information. For example, in one class of gradient-based techniques, the partial derivatives of the objective and constraint functions at the current point are calculated (local information), and the direction of minus the gradient of the objective function (or the reduced gradient of the objective function, if there is any equality constraint) then is used to obtain the next point while satisfying the constraints. However, the information provided by the gradient is based on the linearization of the functions at the current point (local information), and may not describe the behavior of the functions completely (incomplete information). Moreover, performance of the optimization technique may be deteriorated by poor selection of various parameters, e.g., difference parameters for numerical differentiation, penalty parameters for penalty methods, initial multipliers for multiplier methods, etc. Therefore, while the algorithm adequately solves one class of problems, unless there is extensive tuning by the user on the algorithm it may perform poorly on others. This difficulty has led to a general belief the industry that only experts in design optimization can apply optimization techniques.

A coupled algorithmic-heuristic approach to the solution of nonlinear programming problems goes beyond the traditional optimization techniques. A description of the approach together with its major features is presented in Appendix A.

D. INFORMATION REQUIREMENTS FOR PRACTICAL APPLICATION

1. Selecting a Method

Once an engineering optimization problem has been formulated, the designer must choose a method of solution. (This section follows Gill, et al. [Ref. 18].) This choice is based on the classification of problems according to the properties of their objective and constraint functions. The selection of the solution method also will depend on other

information available to the engineer, such as primarily knowledge of the derivatives of the problem functions and the availability of appropriate computer software and hardware.

Knowledge of the derivatives of the problem functions, particularly the exact first derivatives, greatly enhances the reliability and decreases the complexity of the optimization methods that may be employed. Similarly, it may not be cost effective to use the Newton-type methods that depend on knowledge of the exact second derivatives when the cost of computing the Hessian matrix is much greater than the cost of computing the gradient. Thus there is a trade-off between the cost of obtaining derivative information and the resulting reliability and accuracy of the solution methods.

a. Unconstrained Problems

For unconstrained problems with a small number of variables, the ranking of methods with respect to the probability of obtaining an acceptable solution is as follows:

- Newton-type with second derivatives
- Newton-type without second derivatives
- Quasi-Newton with first derivatives
- Quasi-Newton without first derivatives
- Conjugate-gradient-type with first derivatives
- Conjugate-gradient-type without first derivatives
- Polytope.[3]

This ranking can be used both for choosing a method to initially attack the problem and for selecting an alternative if the first method does not provide an acceptable solution.

The utilization of second derivatives gives the Newton-type method its top ranking because the second-order properties enhance algorithmic efficiency and yield qualitative data about the resulting solution. The Hessian matrix, evaluated at the solution, can provide estimates of the conditioning and sensitivity of the solution while a quasi-Newton approach may not give an accurate representation of the actual Hessian.

Finite differences of gradients may be used to provide approximations to the second derivatives, which can be very effective in Newton-type methods. However, in quasi-Newton-type methods, the use of finite difference methods will alter choice of step-length

---

[3] See Section I.B of Volume I of this report.

algorithms and change the termination criteria. The polytope method is the most unreliable of the methods since it uses the least information about the objective function.

When the number of variables in an unconstrained optimization problem is large, the approach to determining a solution method is less clear than with the small-scale problems. Both sparse-discrete Newton methods and conjugate-gradient-type algorithms have been used successfully with these larger problems.

b. Linearly Constrained Problems

The methods listed above for unconstrained problems with smooth objective functions can be adapted to linearly constrained problems. The relative ranking of methods for the linear constraint case is the same as the ranking for the unconstrained case.

A significant factor in selecting a method and software to optimize a linearly constrained problem is the size of the problem, i.e., whether it is sparse or dense. The importance of the size factor stems from the historical development of software for the two classes of problems. Software for sparse problems has been well developed commercially, whereas software for dense problems has remained more in the university research setting. This difference in development has resulted in a difference in user friendliness of the two classes of codes, with the commercial codes being easier to handle.

However, numerical methods for handling dense problems have superior properties due to better conditioning of the search direction and Lagrange multiplier estimates. For very dense problems, null-space active-set methods will probably prove best. As the problem increases in size, range-space methods become more attractive.

Often, objective functions are defined only in the feasible region. Outside the feasible region, the values of the objective function may represent situations that have no physical counterpart. The user may specify that the objective function be computed only in the feasible region. To be certain of this, the user, in general must provide exact derivatives. If the problem contains only inequality bound constraints, finite difference approximations to first or second derivatives may be adapted so that the objective function is calculated only in the feasible region.

c. Nonlinearly Constrained Problems

A number of the methods for nonlinearly constrained problems will produce iterations of the objective function that are not necessarily in the feasible region. The user must decide whether feasibility is necessary before choosing a method.

d. Approximations

If a user does not have access to a complete numerical software library or if the library does not contain a routine for the specific problem of interest, then the user must make do with what they have. In general, the user should choose a method that takes maximum advantage of any structure that the problem exhibits.

If some of the first or second derivatives are missing, the user may select a scheme that will approximate the missing derivatives. If the problem is nonlinearly constrained but the available optimization routine is for smooth unconstrained problems, the constrained problem may be transformed into a sequence of unconstrained problems by an augmented Lagrangian method. Least squares methods may be used when there are no available routines for solving nonlinear equations.

2. Assessment of Results

Optimization algorithms are plagued with two major drawbacks: (1) reporting a failure to solve the problem when a solution has been found, and (2) reporting a solution that is not a solution.

This situation makes it incumbent on the user to verify any information that the algorithm returns, with regard to the validity of the solution, whether the information is positive or negative. When a good algorithm satisfies the termination conditions, this should mean that a close approximation to an optimal solution has been reached. However, the user should verify that the algorithm has not terminated prematurely.

a. Unconstrained Problems

There are three conditions that should be checked when an optimization algorithm has terminated at a point x_m:

(1) norm {grad $F(x_m)$} << norm {grad $F(x_o)$}

(2) the iterations preceding x_m exhibit rapid convergence to x_m

(3) the Hessian matrix (or its approximation) at x_m is well-conditioned.

Given that all three conditions are met, x_m is probably near the solution whether or not the algorithm indicated a successful termination.

Condition (1) indicates that the optimization is probably successful if the norm of the gradient of the objective function is much smaller at the candidate solution point than it is at the starting point. This shows that x_m is closer to a saddlepoint than x_o.

Condition (2) is an effective check because many algorithms exhibit a rate of convergence that is faster than linear in the neighborhood of the solution.

Condition (3) is useful when the algorithm displays a linear or sublinear rate of convergence near x_m. This can be due to discontinuities in the second derivatives. If the Hessian is well-conditioned, small discontinuities will probably not significantly inhibit the rate of convergence. Consequently, the discontinuities are likely to be large enough to be located and possibly eliminated.

b. Constrained Problems

Conditions 1-3, given above for unconstrained problems, can be generalized to constrained problems by appropriate mathematical substitutions in both the linear and nonlinearly constrained cases. When there are nonlinear constraints, condition (2) should be modified further to include a check on the rate of convergence of the Lagrange multiplier estimates.

When the Lagrange multipliers are zero or near zero, difficulties arise in judging the validity of the solution. This is because the sign of the Lagrange multiplier for an active constraint is essential in determining whether the candidate solution is optimal and in showing how to move toward the optimum if the candidate is nonoptimal. When the Lagrange multiplier is near zero, the algorithm may activate the wrong constraint and return a candidate solution that is far from optimal.

Other difficulties arise if a Lagrange multiplier is exactly zero. The zero indicates that, to first order, small changes in the constraint do not alter the objective function. Resolution would require examination of higher order derivatives, which may not be available. The zero value of the multiplier sometimes means that the solution would be the same if the constraint was eliminated while in other situations elimination of the constraint corresponding to the zero multiplier does alter the solution.

Good software will attack the near zero multiplier problem by eliminating and/or perturbing the relevant constraints and checking the effect on the objective function.

Large Lagrange multipliers usually have an adverse effect on convergence and thus need to be corrected. They can be corrected if they arise from poor scaling of the constraints. A less likely possibility is present in some nonlinearly constrained problems in which the Jacobian matrix of the active constraints is rank-deficient.

If a check of conditions 1-3 above do not yield convincing indications of the optimality of the candidate solution, three options remain: alter the parameters of the algorithm, employ a different optimization method, or change the problem.

In altering the parameters of the algorithm for a step problem, the easiest parameter to alter is the step length accuracy. An approach for other problems is to start the algorithm at a new initial approximation point that is not too close to the old initial point or any of the points in the previous iterations. There are other parameter alterations which depend on the specific method being used. Convergence to the same point after a parameter alteration gives some increase in confidence that the point is the solution. Yet, when second derivatives are not available, there may be repeated convergence to a point that has a very small gradient, yet is not a solution.

The next step toward verification is using an alternative method and starting from a different point. If even this does not work, the last chance is to reformulate the problem in the hope that the difficulty was due to an improper formulation.

c. Performance

Performance of optimization algorithms can be determined in two basic ways. One is by theoretical analysis to check convergence rates and computational complexity. The second is by actual testing of an implementation of the algorithm on a set of benchmark problems. Theoretical approaches only yield estimates of real world performance, so computational benchmarks are preferred. User-friendliness can only be assessed through actually using the code in question.

The Committee on Algorithms (COAL) of the Mathematical Programming Society and the National Bureau of Standards have led the effort to establish sound experimental techniques to test mathematical programming codes. The performance measures used include efficiency, user-friendliness, generality, robustness, capacity, and programming quality.

E. CURRENT CAPABILITIES OF OPTIMIZATION SOFTWARE

1. Computing Advances

The advancing technologies in the computer hardware and software industries are continuing to have a major impact on applications of mathematical optimization methods. (This section follows Waren, et al [Ref. 71]). The increased performance of computers makes it possible to solve additional and larger problems while reducing the associated costs. Also, the advances make it feasible to include optimization routines in large user-friendly application systems that require little familiarity with the optimization methods.

The increasing power and availability of microcomputers and personal engineering workstations has greatly increased the potential applications of mathematical optimization routines. Such machines can readily be used to solve medium sized (< 100 variables) nonlinear problems. Furthermore, such machines are serving as intelligent input/output interfaces to more powerful mainframes and supercomputers. This allows the users to approach problems interactively instead of by batch processing, employ floppy or hard disk files rather than magnetic tapes; see graphical, color-coded results on a graphics monitor; and print or chart only what is really necessary.

Recent years also have brought significant advances in mathematical programming techniques such as in Successive Linear Programming (SLP), Successive Quadratic Programming (SQP), related Lagragian methods, and ellipsoidal approaches to nonlinear programming (NLP) for inequality constrained problems. Another advance has occurred through the creation of modeling systems with built-in NLP routines. Such systems allow the problems to be expressed in spreadsheet format or algebraically or are data driven.

The following sections provide a sample of nonlinear optimization software for various classes of computers and sizes of problems.

2. Mainframe Software for Medium Size Problems

ADS: This is a library of optimization routines which includes penalty, augmented Lagragian, SLP, SQP, and feasible direction methods. Within each algorithmic strategy, the user chooses from several procedures for determining the search direction. ADS uses reverse communication to return control to the user's program whenever new information is needed on function or gradient values. This feature gives ADS great flexibility for revision of strategies as the optimization progresses. ADS also interfaces with

NASTRAN, a finite element analysis code, so that engineers can couple simulation and optimization.

EA3: This employs an ellipsoidal method for solving systems of nonlinear inequalities. It cannot be used for equality constraints. The user is required to provide bounds for all variables and routines for computing function and gradient values. Since no other user input is required, the code is relatively easy to use. Because of the many function evaluations performed by the code, it is somewhat slow. EA3 is written in FORTRAN 77 for IBM and can be ported to other systems.

GRG2: This is an implementation of the Generalized Reduced Gradient algorithm. It is coded in FORTRAN 77 and runs on many different computers. Users are required to provide routines for evaluating objective and constraint functions. The code includes numerical differencing capabilities so that user provision of a partial derivative routine is optional.

NLPQL: This code is based on the Harwell routine, VF02AD, using the SQP method. It is a Fortran subroutine requiring the user to provide a calling program as well as routines to calculate the functions and their gradients.

NOC OPTIMA: This is a library of FORTRAN subroutines for nonlinear optimization problems. The library includes a variety of subroutines including an unconstrained optimizer, a least squares solver, and constrained optimizers using SQP and SUMT. The user is required to provide a calling program, routines for giving the objective function and constraint values, and an optional routine for the derivatives.

OPT: This code is written in FORTRAN 77 and is an implementation of the GRG algorithm. The user provides a calling program that dimensions arrays and fixes tolerances and limits. The user also must give routines for evaluating the objective and constraint functions. A partial derivative subroutine is optional.

SOL/NPSOL: This code is a collection of FORTRAN subroutines implementing an SQP algorithm. The user fixes limits and tolerances, provides a calling program, a routine to evaluate the nonlinear constraints and their derivatives, a routine for evaluating the objective function and its derivatives, and the coefficients of any linear constraints.

VMCOM: Like NLPQL, this routine is based on the Harwell routine VF02AD. The user must provide a routine to give values for the objective function, the constraints, and the gradients.

VMCWD: This is a FORTRAN subroutine based on an SQP algorithm which employs the "watch dog" method. The user provides a routine to evaluate the objective function, the constraints, and the gradients.

3. Mainframe Software for Large Problems

These codes are capable of handling nonlinear problems with thousands of constraints and variables.

CONOPT: This code is a sparsity-oriented implementation of the GRG algorithm, with capabilities for dealing with multiperiod problems. It includes a sophisticated control language for setting parameters and options, dealing with errors, and checking for model errors.

MINOS 5.0: This is the most widely used code for large scale nonlinear problems. It attacks large space NLP's through a sequence of linearly constrained problems. Each problem in this sequence uses constraints that are linearized versions of the original constraints. The nonlinear objective function is transformed into an augmented Lagrangian.

SCICONIC: This code employs an SLP method and the variable reduction concept of the GRG algorithm. The nonlinear problem is formulated in such a way that the variables are divided into three classes: nonlinear variables, linear variables with variable coefficients, and linear variables with constant coefficients. At each iteration, they are divided further into dependent and independent classes. The independent nonlinear variables are varied with a conjugate gradient algorithm and the dependent variables are calculated using a linear programming approximation.

SLP: This uses a Successive Linear Programming algorithm with a trust region strategy employing an exact penalty function. It is particularly effective on problems with vertex or near-vertex optima. Because it uses the method of steepest descents, it tends to be slower than other codes in the case of numerous degrees of freedom at the optimum and a state of well-conditioning.

4. Software for Microcomputers

GINO: This is an interactive nonlinear optimizer for the IBM PC and compatibles. Its brother LINDO is a linear optimizer. GINO uses GRG2. For a PC with 256Kb memory the problem size limit is 30 constraints and 50 variables as well as their bounds.

NLPROLOG: This IBM PC code is part of a set of programs for solving mathematical programming problems. It uses a fixed point method for nonlinear problems with provision for both differentiable and non-differentiable functions. The user provides the functions and constraints through a BASIC text file or via FORTRAN subroutines. A PC with 512Kb can handle problems with a maximum total of 150 variables and constraints.

NLP SOLVER: This IBM PC code is based on the SQP method and can handle a problem with 25 variables and 50 constraints in 256Kb memory.

OPTISOLVE: This IBM PC code uses a penalty method to transform nonlinearly constrained problems into sequences of unconstrained problems which are solved with a variable metric method. After a solution is returned, the code provides a performance index between 0 and 100 to gauge the accuracy of the solution.

5. Modeling Systems

EMP: This system requires the user to be familiar with FORTRAN. It is used to solve general nonlinear problems. EMP uses two solvers--NLPQL and an ellipsoidal method similar to EA3. The limit on the problem size is a total of 100 variables and constraints. It is a menu driven, interactive system that prompts the user for further information. EMP includes facilities organizing problem specifications and results, which include editing, adding, deleting, displaying, and sorting.

GAMS: The General Algebraic Modeling System organizes an optimization model as several sets of equations. These equations are set up using an algebraic syntax that allows the user to specify an indexed set of equations with a single GAMS statement. GAMS uses NPSOL and MINOS 5.0 as its NLP solvers.

IFPS/OPTIMUM: This system employs a nonprocedural spreadsheet language with each cell containing a rule for determining the value of the variable in that cell. The problem is set up by specifying which cells contain the decision variables, the objective, and the constraints. The user also provides bounds on the decision variables and constraints. The system is able to determine whether the problem is linear or nonlinear, and sends this information to the appropriate solver. The solvers include GRG2 for nonlinear problems, XMP for linear problems, and ZOOM/XMP for linear mixed integer problems.

F. RESEARCH NEEDS

In the following sections, current and future research areas in design optimization, in particular the numerical methods, will be discussed. These sections are prepared based on a report of the SIAM Activity Group on Optimization that was recently published in the SIAM News (March 1987). The final section targets research areas in design optimization for tradeoffs encountered in ULCE.

1. Current Research

Despite the fact that a wide variety of optimization methods have been developed over the last decade, there a considerable gap exists between the breadth of the available theory and the demands of routine application to meaningful engineering design problems.

Current research in numerical optimization techniques concentrates on the unconstrained and constrained methods. As progress has been made over the last decade in understanding the unconstrained methods, the research has focused progressively on constrained optimization. However, advances in constrained optimization have been minimal compared with those for unconstrained. As advances in modeling techniques and computer technology allow industry to consider progressively more complicated models, the need for advances in the constrained optimization techniques becomes increasingly important.

The major part of research activities in unconstrained problems is on quasi-Newton or variable metric methods in that they are based primarily on the properties of the quadratic functions. In particular, the methods and convergence theories for BFGS and DFP methods for unconstrained optimization are now well established and are becoming, more or less, classic. The local convergence theory for the BFGS and DFP methods requires that the Hessian matrix of the function being minimized be positive definite at the solution. However, the global convergence theory for the BFGS and DFP methods does not seem to be as well understood.

In extending quasi-Newton methods to constrained problems, the formulation of the model subproblem presents no difficulty. It is a quadratic program in which the Hessian of the quadratic objective function is the Hessian of the Lagrangian or a secant approximation to the Hessian. This approach is the sequential quadratic programming (SQP) approach to constrained optimization, as explained in Section 4.b. The difficulty in extending the quasi-Newton methods to constrained problems arises from the requirement

of the local convergence theory for the BFGS and DFP: the Hessian of the Lagrangian must be positive definite at the solution.

In unconstrained optimization, the standard assumptions coupled with the second order necessary conditions produce positive definiteness. In the case of constrained optimization, however, this coupling gives positive definiteness only on a proper subspace. It follows that an approximation of the Lagrangian may not be able to be updated. Thus, the constrained algorithm is not well defined. Numerous modifications have been suggested for this problem, but none has been supported yet by convergence theory.

A major problem in constrained optimization is the choice of the merit function. The merit function is defined as the function used to decide whether one approximate solution is better than another. In unconstrained optimization the obvious choice is the objective function. However, in constrained optimization, the merit function should reflect the objective function and the constraints' infeasibilities. Choosing a merit function that accomplishes this objective and meshes well with the direction calculated from the quadratic programming subproblems is not an easy task, and the optimal one is yet to be found.

One area of research in constrained optimization that has no counterpart in unconstrained optimization is the implementation of inequality constraints. Inequality constraints have traditionally been handled by adjoining a squared slack variable to each constraint. The design optimization problem then is formulated as an equality constrained problem. If the SQP method is then applied, the quadratic programming subproblem will have only equality constraints. The problem can then readily be solved as a system of linear equations. Although this approach is used successfully in many engineering design problems, critics of this approach point out that not only does the dimension increase, but also that the numerical conditioning deteriorates.

Two other popular approaches for handling inequality constraints in numerical optimization are active set strategies and linearization of constraints. The active set strategies divide the inequality constraints at each iteration into those that will be treated as equality constraints and those that will be ignored. One way of accomplishing this is to treat constraints that are violated as equalities and to ignore those that are most satisfied. Another way is to pick the active constraints based on the information provided by Lagrange multipliers or other means. If the rules that are used to decide which constraints should be considered active or ignored are simple, then "cycling" or "zigzagging" may occur and convergence may be lost. In the linearization of constraints approach, the

inequality and/or equality constraints are linearized by expanding them about the current point using the first-order Taylor series.

Some research activity also has been directed toward solution strategies that correctly identify the active constraints. Under certain assumptions, the SLP and SQP methods have this desirable property.

2. Future Research

Problems that warrant additional research activity are those that have not yet been solved for unconstrained, let alone for constrained, optimization. Included in this category are algorithms for design problems in which the derivatives do not exist, for extremely large problems, and for algorithms that can be efficiently implemented on multiprocessor computers. The latter include multilevel decomposition methods applicable to a general class of nonlinear design problems.

Since solution strategies for nonlinearly constrained optimization problems generally result in a local rather than a global solution, research activity should be directed toward global optimization methods. Current tools used in these methods are based on such diverse areas as the physics of the problem, probability theory, and interval analysis.

The greatest activity in numerical optimization research will most likely continue around Karmarkar's method for linear programming [Ref. 33]. Generalization of Karmarkar's method to quadratic programming problems has received some attention, but the proper generalization has not yet been found. Three possible research directions might emerge. The first is a direct extension of Karmarkar's method to general nonlinearly constrained design optimization problems. The second is in the development of an SLP approach to nonlinear programming with Karmarkar's method used for the linear programming subproblem. The third direction is an extension of Karmarkar's method to quadratic programming with the development of a nonlinear programming method.

3. Importance of Research in Design Optimization for ULCE

Design optimization is at a rather critical phase because there is a need to carefully examine its present conditions and future prospects. Although this volume of the report has dealt thoroughly with mathematical and analytical techniques, the primary concern for ULCE is to determine how optimization fits into the overall design process. The design process itself is a highly interactive, coupled activity that requires quantitative, systematic,

and analytical methods at all levels, from the identification of the need to the final testing and evaluation. A systematic method for design should involve (see, for example, "Goals and Priorities for Research in Engineering Design" [Ref. 72]):

- Consistent multilevel representation of the systems being designed.
- Effective communication with the multilevel simulations developed for a design problem.
- Decomposition or partitioning of a design problem into manageable segments without seriously affecting the integrity of the total problem.
- Quantitative assessment of the manufacturability, reliability, controllability, and maintainability of a product.
- Mechanisms that expedite the search through exploration of the design alternatives.

The emphasis for ULCE should be on the basic need that optimization by numerical algorithms and other methods must be integrated within the whole design process. Of great importance are the development and application of optimization techniques that help to systemize the trade-offs that are involved in the design of the complex systems, particularly the following.

- Decomposition-based optimization of engineering design problems should have a high priority. Further, methods that can handle the hierarchical systems with various structures, configurations, and interactions are highly desirable.
- Methods that allow explicit treatment of realistic constraints, as opposed to implicit penalty functions, as well as the use of cost-objective functions, which are more directly relevant to the performance of the systems, is needed.

These issues are explored further and a plan for their research and development for ULCE application is outlined in Volume I of this report--An Evaluation of Potential Research Directions.

REFERENCES

1. Papalambros, P., D. J. Wilde, *Principal of Optimal Design - Modelling and Computation,* Cambridge University Press, New York, 1987 (in press).

2. Siddall, J. N., *Optimal Engineering Design,* Marcell Dekker Inc., New York, 1982.

3. Wismer, D. A., *Optimization Methods for Large-Scale Systems,* McGraw-Hill Co., 1971.

4. Bellman, R. E., *Dynamic Programming,* Princeton University Press, 1957.

5. Johnson, R. S., and R. C.Benson, A Basic Two-Stage Decomposition Strategy in Design Optimization, *ASME Trans., Journal of Mechanisms, Transmissions, and Automation in Design,* Vol. 106, pp. 380-386, 1984a.

6. Johnson, R. S., and R. C.Benson, "A Multistage Decomposition Strategy for Design Optimization," *ASME J. of Mechanisms, Transmissions, and Automation in Design,* Vol. 106, pp. 387-383, 1984b.

7. Dantzig, G. B., and P. Wolfe, "Decomposition Principle for Linear Programs," *Operations Research* 8, pp. 101-111, 1960.

8. Lasdon, L. S., *Optimization Theory for Large Systems*, Macmillan Co., London, 1970.

9. Kirsch, U., *Optimum Structural Design*, McGraw-Hill, N.Y., 1981.

10. Reklaitis, G. V., A. Ravindran, and K. M. Ragsdell, *Engineering Optimization,* John Wiley and Sons, New York, 1983.

11. Apostol, T. M., *Calculus, Vol. II,* Wiley, New York, 1969.

12. Wilde, D. J., and C. Beightler, *Foundations of Optimization,* Prentice-Hall, N.J., 1967.

13. Karush, W., "Minima of Functions of Several Variables with Inequalities as Side Conditions," MS Thesis, Dept. of Mathematics, University of Chicago, 1939.

14. Kuhn, H. W., and A. Tucker, "Nonlinear Programming," Proceeding of the Second Symposium on Mathematical Statistics and Probability, University of California Press, Berkeley, 1951.

15. Avriel, M., *Nonlinear Programming*, Prentice-Hall, N.J., 1976.

16. Bazaraa, M. S., and C. M. Shetty, *Nonlinear Programming*, Wiley, N.Y., 1979.

17. Powell, M. J. D., "An Efficient Method for Finding the Minimum of a Function of Several Variables Without Calculating Derivatives," *Computer* J., 7, pp. 155-162, 1964.

18. Gill, P. H., W. Murray, and M. H. Wright, *Practical Optimization*, Academic Press, London, 1981.

19. Himmelblau, D. M., *Applied Nonlinear Programming*, McGraw-Hill, N.Y., 1972.

20. Vanderplaats, G. N., *Numerical Optimization Techniques for Engineering Design*, McGraw-Hill, N.Y., 1984.

21. Cauchy, A., "Method generale pour la resolution des systemes d'equations simultanees," *Computer Rend. Acad. Sci.*, 25, pp. 536-538, 1847.

22. Fletcher, R., and C. M. Reeves, "Function Minimization by Conjugate Gradients," *Computer J.*, 7, pp. 149-154, 1964.

23. Davidson, W. C., "Variable Metric Method for Minimization," AEC Res. Develop. Rep., ANL-599, 1959.

24. Fletcher, R., and J. D. Powell, M. "A Rapidly Convergent Descent Method for Minimization," *Computer J.*, 6, pp. 163, 1963.

25. Broyden, G. G., "The Convergence of a Class of Double-Rank Minimization Algorithms," *J. of Inst. Math. Appl.*, 6, 76-90, 222-231, 1970.

26. Fletcher, R., "A New Approach to Variable Metric Algorithms," *Computer J.*, 13, pp. 317-322, 1970.

27. Goldfarb, D., "A Family of Variable Metric Methods Derived by Variational Means," *Math. Comput.*, Vol. 24, pp. 23-36, 1970.

28. Shanno, D. F., "Conditioning of Quasi-Newton Methods for Function Minimization," *Math. Comput.*, Vol. 24, pp. 647-656, 1970.

29. Sargent, R. W. H.and D. J. Sebastian, "Numerical Experience with Algorithms for Unconstrained Minimization," *Numerical Methods for Non-Linear Optimization* (F. A. Lootsma, Ed.), Academic Press, N.Y., Chap. 5, 1971.

30. Dantzig, G. B., "Programming in a Linear Structure," Comptroller, Wash. D.C., Feb. 1948.

31. Fabozzi, E. J.and J. Valente, "Mathematical Programming in the American Companies: A Sample Survey," *Interfaces*, 7(1), pp. 93-98, 1976.

32. Steen, L. A., "Linear Programming: Solid New Algorithm," *Science News*, 116, pp. 234-236, 1979.

33. Karmarkar, N., "A New Polynomial-Time Algorithm for Linear Programming," Proceedings of the 16th Annual ACM Symposium on the Theory of Computing, pp. 302-311, 1984.

34. Powell, M. J. D., "Optimization Algorithms in 1979," Committee on Algorithms Newsletter, No. 5, Mathematical Programming Society, pp. 2-16, Feb. 1981.

35. Kelly, J. E., "The Cutting Plane Methods for Solving Convex Programs," *SIAM J.*, 8, pp. 703-712, 1960.

36. Murty, K., Linear and Combinatorial Programming, Wiley, 1976.

37. Powell, M. J. D., "A Fast Algorithm for Nonlinearly Constrained Optimization Calculations," DAMPTP77/NA 2, University of Cambridge, England, 1977.

38. Sandgren, E., "The Utility of Nonlinear Programming Algoritms," PhD Thesis, Dept. of Mechanical Engineering, Purdue Univ., 1977.

39. Hock, W. and K. Schittkowski, Test Examples for Nonlinear Programming Codes, Springer-Verlag, New York, 1980.

40. Rosen, J. B., "The Gradient Projection Method for Nonlinear Programming," Part I, Linear Constraints, *SIAM J. of Applied Mathematics*, Vol. 8, pp. 181-217, 1960.

41. Zoutendijk, G., Methods of Feasible Directions, Elsevier, Amsterdam, 1960.

42. Topkis, D. M.and A. F. Veinott, "On the Convergence of Some Feasible Direction Algorithms for Nonlinear Programming," *SIAM J. Control*, 5, pp. 268-279, 1967.

43. Wolfe, P., "Methods of Nonlinear Programming," *Recent Advances in Mathematical Programming* (R. Graves and P. Wolfe Eds.), 1963.

44. Abadie, J., and J. Carpentier, "Generalization of the Wolfe's Reduced Gradient Method to the Case of Nonlinear Constraints," *Optimization* (R. Fletcher Eds.), 1969.

45. Gomory, R. E., "An Algorithm for Mixed Integer Problem," *Rand Report*, R.M. 25797, 1960.

46. Land, A. H., and A. Doig, "An Automatic Method for Solving Discrete Programming Problems," *Econometrica*, Vol. 28, pp. 497-520, 1960.

47. Balas, E., "An Additive Algorithm for Solving Linear Programs With Zero-One Variables," *Operations Research*, 13, pp. 517-546, 1965.

48. Gelattly, K. M. and P. B. Marcal, "Investigation of Advanced Aircraft Technology," *NASA Report* No. 2356-950001, 1967.

49. Gisvold, K. M.and J. Moe, "A Method for Nonlinear Mixed-Integer Programming and its Application to Design Problems," ASME J. of Engineering for Industry, Vol. 94, pp. 353-364, 1972.

50. Wilde, D. J., Globally Optimal Design, Wiley, N.Y., 1978.

51. Wilde, D. J., "Monotonicity and Dominance in Hydraulic Cylinder Design," *ASME J. of Engineering for Industry*, Vol. 94, No. 4, 1975.

52. Papalambros, P.and D. J. Wilde, "Global Non-Iterative Design Optimization Using Monotonicity Analysis," *ASME J. of Mechanical Design*, Vol. 101, No. 4, pp. 643-649, 1979.

53. Papalambros, P.and D. J. Wilde, "Regional Monotonicity in Optimum Design," *ASME J. of Mechanical Design*, Vol. 102, No. 3, pp. 497-500, 1980.

54. Zhou, J., R. and W. Mayne, "Interactive Computing in the Application of Monotonicity Analysis to Design Optimization," *ASME J. of Mechanisms, Transmissions, and Automation in Design*, Vol. 105, No. 2, pp. 181-186, 1982.

55. Azarm, S.and P., Papalambros, "An Automated Procedure for Local Monotonicity Analysis," *ASME J. of Mechanisms, Transmissions, and Automation in Design*, Vol. 106, No. 1, pp. 82-89, 1984.

56. Zhou, J.and R. W. Mayne, "Monotonicity Analysis and the Reduced Gradient Method in Constrained Optimization," *ASME J. of Mechanisms, Transmissions, and Automation in Design*, Vol. 106, No. 1, pp. 90-94, 1984.

57. Davis, W. J., "A Generalized Decomposition Procedure and its Application to Engineering Design," *J. of Mechanical Design*, Vol. 100, pp. 739-746, 1978.

58. Siddall, J. N. and W. K. Michael, "Large Systems Optimization Using Decomposition with Soft Constraints," *ASME J. of Mechanical Design*, Vol. 102, pp. 506-509, 1980.

59. Haftka, R. T., "An Improved Computational Approach for Multilevel Optimum Design," *ASME J. of Structural Design*, 12(2), pp. 245-261, 1984.

60. Sobieski, J., "A Linear Decomposition Method for Large Optimization Problems," NASA TM 83248, NASA Langley, Hampton, VA, 1982.

61. Sobieski, J.,J.-F. Barthelemy, and G. I. Giles "Aerospace Engineering Design by Systematic Decomposition and Multilevel Optimization," 14th Congress of the International Congress of the Aeronautical Sciences, Paper No. ICAS-84-4.7.3, 1984.

62. Sobieski, J.and R. T. Haftka, "Interdisciplinary Optimum Design," *Proceedings of the NATO Advanced Study Institute in Computer-Aided Optimal Design*, Portugal, pp. 29-60, 1986.

63. Kron, G., "A Method for Solving a Very Large Physical System in Easy Stages," *Proceedings of the Inst. of Radio Engineers*, Vol. 42, No. 4, pp. 680-686, 1954.

64. Azarm, S.and W. Li, "A Two-Level Decomposition Method for Design Optimization," invited paper in the Proceedings of an NSF workshop in Design Theory and Methodology, pp. 453-502, Feb. 1987 (also under review in Engineering Optimization Journal).

65. Azarm, S.and W. Li, "Optimal Design Using a Two-Level Monotonicity-Based Decomposition Method," *ASME Design Automation Conference,* 1987 (under review in ASME Journal of Mechanisms, Transmissions, and Automation in Design).

66. Arora, J. and G. Baenziger, "Uses of AI in Design Optimization," *Computer Methods in Applied Mechanics and Engineering,* Vol. 54, pp. 303 (1986).

67. Azarm, S. and M. Pecht, "Knowledge Gathering For Heuristic Programming in Design Optimization," Accepted for publication: *Journal of Engineering Optimization,* Jan. 10 (1987).

68. Azarm, S. and M. Pecht, "A Coupled Algorithmic-Heuristic Approach For Design Optimization," *IEEE Transaction on Systems, Man and Cybernetics,* Vol. 17, May (1987).

69. Azarm, S. and P. Papalambros, "A Case Study for a Knowledge-Based Active Set Strategy," *Journal of Mechanical Transmissions and Automation in Design,* Vol. 106, No. 1 (1984).

70. Li, H. and P. Papalambros, "A Production System for Use of Optimization Knowledge," *Journal of Mechanical Transmissions and Automation in Design.*

71. Waren, A.D.and M.S. Hung, and Lasdon, L.S., "The Status of Nonlinear Programming Software: An Update," 1986, to be published.

72. "Goals and Priorities for Research in Engineering Design," A Report to Design Research Community, *The American Society of Mechanical Engineers,* July 1986.

Appendix A

A COUPLED ALGORITHMIC-HEURISTIC OPTIMIZATION SYSTEM

A COUPLED ALGORITHMIC-HEURISTIC OPTIMIZATION SYSTEM

A. INTRODUCTION

The need to develop optimization techniques that can use information not used by traditional nonlinear programming techniques was discussed first by Azarm and Papalambros [Refs. 55, 69] and implemented in a production system by Li and Papalambros [Ref. 71]. In that research, the idea was to integrate global knowledge available for a particular problem into a nonlinear programming technique. The knowledge was organized in the form at of rules describing possible constraint activity or inactivity, redundancy, and dominance. Motivation for that research came mainly from experiences with monotonicity analysis.

Here, a different approach for building an optimization system is proposed. This idea is to use different local optimization strategies and to observe the effect of each strategy on the overall performance of the NLP method by using a set of test problems. In this study, the parameters used to measure the overall performance of the NLP method include the number of objective function evaluations, the number of constraint functions evaluations, and the number of gradient evaluations. The observations made here become the knowledge utilized by the heuristic procedures. The heuristic procedures essentially select the local optimization strategy that is best suited to solve a particular problem based on the overall performance. Since the action taken by heuristic procedures may not be deterministic, the strength of their action is examined based on the degree of certainty in their premise.

B. DESCRIPTION OF THE OPTIMIZATION ALGORITHM

The general optimization problem is formulated as:

minimize $f(\mathbf{x})$

subject to:

$g_j(\mathbf{x}) = 0 \qquad j = 1, \ldots, m$

$g_j(\mathbf{x}) =< 0 \qquad j = (m+1),\ldots,p$

where f and g_j are scalar objective and constraint functions, and $\mathbf{x}$ is a n-vector of design variables.

The optimization algorithm described here is based on the observation that in design optimization there usually exists a large number of inequality constraints, many of them satisfied as equalities at the optimum (active constraint). An active set strategy based on local monotonicity information is then utilized. Two principles used in the local monotonicity analysis are repeated here for convenience. Referring to problem (1) we have the following principles:

(1) If the objective function is monotonic with respect to (w.r.t.) a particular variable in the neighborhood of a local minimum, then there exists at least one active constraint with opposite monotonicity w.r.t. that variable in that neighborhood.

(2) If the objective function is stationary w.r.t. a particular variable in the neighborhood of a local minimum, then either all constraints containing that variable are inactive, or there exists at least two active constraints having opposite monotonicity w.r.t. that variable in that neighborhood.

These principles can be viewed as a special case of the Karash-Kuhn-Tucker (KKT) optimality conditions. Since both principles identify the candidate active constraints, a selection criterion is necessary. The selection criterion uses a local dominance criterion to select the active constraint per rule in a given iteration. If the local prediction of monotonicity is untrue, corrective action, such as a line search between the points generated by two consecutive iterations is taken. The basic steps of the approach are summarized below.

Given an initial point as the current point x;

Step 1:

Find partial derivatives of the objective and constraint functions. If there are some constraints active at this point, then the partial constrained derivatives are evaluated [Ref. 6].

Step 2:

If $||\nabla f(x)|| \leq \varepsilon$

and if

(a) x is feasible, then check KKT optimality conditions. If they are satisfied, then $x = x^*$ and stop; otherwise deactivate constraint(s) with negative Lagrange multiplier(s) and go to step 1:

(b) x is infeasible, then deactivate the current active set and go to Step 1. Otherwise, continue to Step 3.

Step 3:

In the objective function, select the variable (referred to as the active variable) for which the objective function has the largest absolute partial derivative. Continue to Step 4.

Step 4:

Apply first and second monotonicity principles to identify the active constraint(s). If no constraint is active, go to Step 5; otherwise go to Step 6.

Step 5:

Move along a descent direction to a new point and then go to Step 1.

Step 6:

If estimated monotonicities are preserved, go to Step 1. Otherwise deactivate the constraints associated with offending monotonicities. If monotonicity estimates generate violations pertaining to the objective function, do a one-dimensional search. If the violations pertain to the constraints, make a descent move. Then return to Step 1.

To improve the reliability of the algorithm, a sequential quadratic programming (SQP) technique similar to that suggested by Powell [Ref. 37] can be introduced. Transition from a local monotonicity strategy to the sequential quadratic programming technique occurs whenever there is no improvement in the objective function value after a specified number of iterations. The sequential quadratic programming solves a quadratic programming subproblem in each iteration. This subproblem is an approximation of the Lagrangian subject to linearized constraints of (1), and it is guaranteed to have a positive definite Hessian. The subproblem is stated in the following form:

minimize

$$Q(\delta) = f(\bar{x}) + \delta^t \nabla f(\bar{x}) + \frac{1}{2}\delta^t B\,(\bar{x}, \overline{\lambda})\ \delta$$

subject to:

$$\nabla g_j^t\,(\bar{x})\ \delta + g_j\,(x) = 0,\ j = 1, \ldots, m$$

$$\nabla g_j^t\,(\bar{x})\ \delta + g_j\,(\bar{x}) \le 0,\ j = (m+1, \ldots, p)$$

where

$$\delta = x - \bar{x}$$

$$B \cong \nabla_{**} L\,(\bar{x}, \overline{\lambda})$$

$$L\,(\bar{x}, \overline{\lambda}) = f(\bar{x}) + \sum_{j=1}^{P} \lambda_j g_i\,\bar{x}$$

The solution of this quadratic programming subproblem estimates the Lagrange multipliers and determines the direction of search used in a subsequent one-dimensional search. This one-dimensional minimization has two goals--to decrease the objective function and to minimize the constraint infeasibilities. The function used for one-dimensional minimization is:

$$\Phi(\alpha) = f(x) + \sum_{j=1}^{m} u_j\,|g_j(x)| + \sum_{j=m+1}^{p} u_j\,|\ \max\,(0, g_j\,(x))$$

where

$$x = \bar{x} + \alpha\delta^i,\ u_j \ge 0$$

Here, we select $u_j = |\lambda_j^i|$ for the first iteration and

$$u_j = \max\ \left[\,|\lambda_j^{\,i}|,\ \frac{1}{2}\,(u_j^{\,i-1} + |\lambda_j^{\,i}|)\right]$$

for subsequent iterations to guarantee convergence.

Appendix B

ADDITIONAL READING ON OPTIMIZATION

ADDITIONAL READING ON OPTIMIZATION

BOOKS:

Angel, E. and R. Bellman, *Dynamic Programming and Partial Differential Equations*, Academic Press, NY (1972).

Aris, R., *Discrete Dynamic Programming*, Ginn-Blaisdell Waltham, Mass. (1964).

Arrow, J., "Social Choice and Individual Values," Cowles Commission Monograph 12, Wiley, NY (1951).

Avriel, M., M.J. Rijckaert, and D.J. Wilde, *Optimization and Design*, Prentice-Hall, Englewood Cliffs, NJ (1973).

Aris, R., *The Optimal Design of Chemical Reactors*, Academic Press, New York (1964).

Bagchi, A. and H.T. Jongen, (Eds.), *Systems and Optimization*, Springer-Verlag, Berlin (1985).

Beightler, C.S., D.T. Phillips, and D.J. Wilde, *Foundations of Optimization*, Prentice-Hall, NJ (1979).

Bensoussan, A. and J.L. Lions, (Eds.), *Analysis and Optimization of Systems*, Springer-Verlag, Berlin (1984).

Bracken, J. and G.P. McCormick, *Selected Applications of Nonlinear Programming*, Wiley, New York (1968).

Brayton, R.K. and R. Spence, *Sensitivity and Optimization*, Elsevier Scientific Publishing Co., Amsterdam (1980).

Breuer, M.A., *Design Automation of Digital Systems*, Prentice-Hall, NJ (1972).

Bunn, D.W., *Analysis for Optimal Decisions*, Wiley, Chichester (1982).

Carmichael, D.G., *Structural Modelling and Optimization*, Halsted Press, New York (1982).

Coleman, A., *Game Theory and Experimental Games*, Pergamon, NY (1982).

Dym, C.L. (Ed.), *Application of Knowledge-Based Systems to Engineering Analysis and Design*, Publication No. AD-10, American Society of Mechanical Engineers, New York (1985).

Evtushenko, Y.G., *Numerical Optimization Techniques*, Optimization Software, NY (1985).

Fletcher, R., *Practical Methods of Optimization*, Wiley, NY (1980).

Fox, R. L., *Optimization Methods for Engineering Design*, Addison Wesley, Reading, PA (1971).

Gero, J. (Ed.), *Knowledge Engineering in Computer-Aided Design*, North-Holland, Amsterdam (1985).

Haimes, Y.Y.and V. Chankong, *Decision Making With Multiple Objectives*, Springer-Verlag, NY (1985).

Haug, E.J. and J.S. Arora, *Applied Optimal Design*, Wiley-Interscience, New York (1979).

Hiriart-Urruty, J.B., W. Oettli, and J. Stoer, (Eds.), *Optimization Theory and Algorithms*, Dekker, NY (1983).

Jelen, F.C., *Cost and Optimization Engineering*, McGraw-Hill, New York (1970).

Johnson, R.C., *Optimum Design of Mechanical Elements*; Wiley-Interscience, New York (1961, 1980).

Jones, A.J., *Game Theory: Mathematical Models of Conflict*, Wiley, NY (1980).

Leitman, G., *Optimization Techniques with Applications to Aerospace Systems*, Academic Press (1962).

Lev, O.E. (Ed.), *Structural Optimization - Recent Developments and Applications*, ASCE, New York (1981).

McCormick, G.P., *Nonlinear Programming*, Wiley (1983).

Mickle, M.H. and T.W. Sze, *Optimization in Systems Engineering*, International Textbook, PA (1972).

Morris, A.J., *Foundations of Structural Optimization: A Unified Approach*, Wiley, Chichester (1982).

Rubinstein, M. F., *Patterns of Problem Solving*, Prentice-Hall, NJ (1975).

Samad, T., *A Natural Language Interface for Computer-aided Design*, Kluwer, Norwell, MA (1986).

Sengupta, J.K., *Optimal Decisions Under Uncertainty*, Springer-Verlag, Berlin (1985).

Serafini, P. (Ed.), *Mathematics of Multi-Objective Optimization,* Springer-Verlag, New York (1985).

Siddall, J.N., *Analytical Decision-Making in Engineering Design,* Prentice-Hall, Englewood Cliffs, NJ (1972).

Spunt, L., *Optimum Structural Design.* Prentice-Hall, Englewood Cliffs, NJ (1971).

Stark, R.M. and R.L. Nichols, *Mathematical Foundations of Design: Civil Engineering Systems,* McGraw-Hill, New York (1972).

Steuer, R.E., *Multiple Criteria Optimization: Theory, Computation, and Application,* Wiley, New York (1986).

Stoecker, W.F., *Design of Thermal Systems*, McGraw-Hill, New York (1971, 1981).

Swap, W.C. (Ed.), Group Decision Making, Sage Publications, Beverly Hills (1984).

Szidarovszky, F., M.E.,Gershon, and L. Duckstein, *Techniques for Multiobjective Decision Making in Systems Management,* Elsevier, New York (1986).

Tillman, F.A., C.L. Hwang, and W. Kuo, *Optimization of Systems Reliability;* Marcel Dekker, N.Y. (1980).

Williams, H.P., *Model Building in Mathematical Programming,* Wiley-Interscience, Chichester (1978).

Yu, P.L., *Multiple-Criteria Decision Making: Concepts, Techniques, and Extensions,* Plenum Press, NY (1985).

Zener, C., *Engineering Design by Geometric Programming* (1971).

PAPERS:

Amyot, J.R., "PAROPT: A Parameter Optimization Program", *ASME Computers in Engineering,* Vol. 1, 1985 .

Azarm. S., "Local Monotonicity in Optimal Design, PhD Dissertation", Department of Mechanical Engineering and Applied Mechanics, University of Michigan, Ann Arbor, 1984.

Betts, J.T., "Frontiers in Engineering Optimization", *Journal of Mechanisms, Transmissions, and Automation in Design,* Vol. 105, June 1983.

Chalfan, K.M., "A Knowledge System that Integrates Heterogeneous Software for a Design Application" *AI Magazine,* Summer 1986, pp. 80-84.

Greene, D.C. and E.E. Lowery, "Supportability Control Factors", *Proceedings Annual Reliability and Maintainability Symposium,* 1986.

Howe, A., J.R. Dixon, P.R. Cohen, and M.K. Simmons, DOMINIC: A Domain Independent Program for Mechanical Engineering Design, *Proceedings First International*

Conference on Application of Artificial Intelligence to Engineering Problems, Southampton, England, April, 1986.

Lilijeqvist, P., J. Nilsson, and L. Sjodin, "Operational and Logistic Resource Optimization at Organizational Level", *Annals of the Society of Logistics Engineers,* October 1986, p.26.

Luksan, L., "A Compact Variable Metric Algorithm for Nonlinear Minimax Approximation", *Computing,* Vol. 36, pp. 355-373, 1986.

Mittal, S., C.L. Dym, and M. Marjaria, "PRIDE: An Expert System for the Design of Paper Handling Systems", *Computer,* July 1986, p. 102

Mostow, J., "Towards Better Models of the Design Process", *AI Magazine,* Vol. 6, No. 1, 1985

Naft, J. and M. Pecht, "A RAMCAD/ULCE Workstation Shell Structure", *Proceedings Annual Reliability and Maintainability Symposium,* 1987 .

Nye, W.T. and A.L. Tits, "An Application Oriented Optimization-Based Methodology of Interactive Design of Engineering Systems", *International Journal of Control,* 1986, Vol. 43, No. 6, 1693-1721.

Penot, J. and A. Sterna-Karwat, "Parametrized Multicriteria Optimization: Continuity and Closedness of Optimal Multifunctions", *Journal of Mathematical Analysis and Applications,* Vol. 120, No. 1, pp. 150-168, Nov. 1986.

Pinter, J., "Extended Univariate Algorithms for n-Dimensional Global Optimization", *Computing,* Vol. 36, pp. 91-103, 1986.

Popplestone, R. et al, "Engineering Design Support Systems", *First International Conference on Applications of Artificial Intelligence in Engineering,* 1986.

Tits, A.L. and Z. Ma, "Interaction, Specification Refinement, and Tradeoff Exploration in Optimization-Based Design of Engineering Systems", *IFAC Control Applications of Nonlinear Programming and Optimization,* Capri, Italy, 1985.

Wang, C., "The Principle and Models of Dynamic Programming", *Journal of Mathematical Analysis and Applications,* Vol. 118, No. 2, pp. 287-308, Sept. 1986.

DISTRIBUTION

IDA PAPER P-2064

DECISION SUPPORT REQUIREMENTS IN A UNIFIED LIFE CYCLE ENGINEERING (ULCE) ENVIRONMENT)

Volume II. Conceptual Approaches to Optimization

63 Copies

	Number of Copies
Department of Defense	
OUSD (R&AT/ET) Rm. 3D1089, Pentagon Washington. DC 20301-3080	
ATTN: Raymond Siewert	1
OUSD (R&AT)/ET Rm. 3D1089, Pentagon Washington, DC 20301-3080	
ATTN: Dr. Leo Young	1
Mr. Russell R. Shorey Assistant Deputy, Systems, OASD/P&L Department of Defense Room 2B322, Pentagon Washington, DC 20301-8000	1
Col. Larry Griffin OASD (P&L) WSIG Rm. 2B322, Pentagon Washington, DC 20301-8000	1
Dr. William E. Isler Director, Prototyping Applications DARPA, ISTO Arlington, VA 22209-2308	1
Office of the Secretary of Defense OUSDRE (DoD-IDA Management Office) 1801 N. Beauregard Street Alexandria, VA 22311	1

Defense Technical Information Center 2
Cameron Station
Alexandria, VA 22304-6145

Miscellaneous, U.S. Government

Dr. Michael J. Wozny 1
Director, Division of Design, Manufacturing,
and Computer-Integrated Engineering
National Science Foundation
1800 G Street, N.W.
Washington, DC 20550

Department of the Army

Mr. Geza Papp 1
Chief of Technology
U.S. Army AMCCOM
Building 62
Picatinny Arsenal
Dover, NJ 07806-5000

Department of the Air Force

Dr. Sam Rankin 1
Director, Mathematical Optimization Program
Air Force Office of Scientific Research
Bolling Air Force Base
Washington, DC 20332-9448

Capt. Maureen Harrington 1
Program Manager, ULCE Decision Support
Logistics and Human Factors Branch
Air Force Human Resources Laboratory
Wright-Patterson AFB, OH 45433-5000

Col. Donald Tetmeyer 1
Director, Logistics and Human Factors Division
Air Force Human Resources Laboratory
Wright-Patterson AFB, OH 45433-5000

Dr. Melvin C. Ohmer 1
Field OPR, ULCE Program
AFWAL Materials Laboratory
AFWAL/CAM
Wright-Patterson AFB, OH 45433

Mr. Nick Bernstein 1
AFWAL/FIBR
Bldg. 45
Wright-Patterson AFB, OH 45433

Capt. John Thomas 1
AFOSR/MM
Wright-Patterson AFB, OH 45433

Mr. James R. Meeker 1
Air Force Systems Command/DLSR
Bldg. 1535, Rm. CD310
Andrews Air Force Base
Washington, DC 20334

Col. Eugene Tatini 1
Air Force Systems Command/PLX
Andrews Air Force Base
Washington, DC 20009

Industrial Organizations

Dr. Shapour Azarm 1
Department of Mechanical Engineering
University of Maryland
College Park, MD 20742

Mr. Mark Erdrich 1
Laboratory Head
Adv. CAD/CAM/CAE
Eastman Kodak Company
7/23 Kodak Park
Rochester, NY 14650

Dr. Iman Foroutan 1
Chief, Engineering Automation Section
Microelectronic Circuits Division
Industrial Electronics Group
Hughes Aircraft Company
P. O. Box H, 500 Superior Avenue
Newport Beach, CA 92658-8903

Mr. Siegfried Goldstein 1
Siegfried Enterprises, Inc.
P. O. Box 2308
North Babylon, NY 11703

Mr. Ken Johnson 1
Manager, Design Technology Department
Lockheed-Georgia Company
Department 72-92/Zone 419
86 S. Cobb Street
Marietta, GA 30063

Dr. Janusz Kowalik 1
Engineering Technology Applications
Boeing Computer Services
2760 160th Avenue, S.E.
Belleview, WA 98008

Dr. Robert Kuenne 1
Professor of Economics
Princeton University
63 Bainbridge Street
Princeton, NJ 08540

Dr. Alan Mitchell 1
Director, Preliminary Design Tool Development
Research and Engineering Division
Boeing Aerospace Company
P. O. Box 3999, MS 82-23
Seattle, WA 98124-5214

Mr. Joseph Naft 1
Director, Computer Aided Design Laboratory
Engineering Research Center
University of Maryland
College Park, MD 20742

Mr. David Owen 1
NTL, Inc.
P. O. Box 597
Northport, NY 11768

Dr. Michael Pecht 1
Associate Professor of Mechanical Engineering
Mechanical Engineering Department
University of Maryland
College Park, MD 20742

Mr. Arne P. Rasmussen 1
Private Consultant
464 Severnside Drive
Severna Park, MD 21146

Mr. Edward Rogan 1
Design Technology Department
Lockheed-Georgia Company
Department D72-92
Marietta, GA 30063

Brother Tom Sawyer 1
Bishop McNamara High School
6800 Marlboro Pike
Forestville, MD 20747-3270

Dr. Edison T.S. Tse 1
Director, Decision Systems Laboratory
Stanford University
Department of Engineering-Economic Systems
Stanford, CT 94305

Institute for Defense Analyses
1801 N. Beauregard Street
Alexandria, VA 22311

Gen. William Y. Smith	1
Mr. Philip L. Major	1
Dr. William J. Schultis	1
Dr. Victor A. Utgoff	1
Dr. Jeffrey H. Grotte	1
Dr. Frederick R. Riddell	1
Mr. William E. Cralley	10
Ms. ML Brei	1
Dr. Karen J. Richter	1
Mr. David A. Dierolf	1
Mr. G. Watts Hill	1
Control and Distribution	10

END

DATE

FILMED

9-88

DTIC